Security Verification and Validation within UK-Based Software SMEs: A Socio-Technical Analysis

Submitted by

Matthew Nicolas Kreeger

for the degree of Doctor of Philosophy

of the

Royal Holloway, University of London

2020

Declaration

I, Matthew Nicolas Kreeger, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed.....(Matthew Nicolas Kreeger)
Date:

Acknowledgements

Although it can sometimes feel like a lonely journey, this research was ultimately supported by many. In particular, I wish to thank my supervisor, Prof. G. Harindranth, for his guidance and support throughout. Likewise, I extend similar appreciation to my advisor, Prof. F. Piper.

Lastly, I extend my heartfelt appreciation, and love, to my family: my wife, my parents and, my son, who, though he joined me part way on the journey, helped me reach the end.

Abstract

Within this thesis we examine how security verification and validation (V&V) is practiced, supported, and perceived, within small to medium-sized software enterprises (SMEs) based within the United Kingdom. Further, we show how the activities associated with this complex socio-technical practice are influenced by the socio-technical characteristics of such organisations - specifically, by determining whether organisational information security culture influences security V&V practice and by establishing whether organisations can possess a mature security V&V practice without a correspondingly mature software V&V practice.

Acknowledging the social-technical nature of V&V, this study is guided and framed through the adoption of the Socio-Technical Interaction Network (STIN) framework. Further, and given that the organisational and technological contexts of this thesis are relatively underexplored, we employ a mixed methods, sequential explanatory design approach towards data collection and analysis. This involved an initial, predominately quantitative web-based survey, with the resultant themes identified being further explored during the subsequent, qualitative semi-structured interviews.

This study contributes to existing empirical software engineering research by highlighting the socio-technical realities surrounding security V&V practice within UK-based software SMEs, hitherto an empirical unknown. Our findings also show that such practice is influenced by an organisation's information security culture and that a relationship between an organisation's software and security V&V practices exists. Notably, we find that software SMEs need to improve their security V&V practice. Whilst this may have been inferred by the increasing number of vulnerabilities being reported, our study highlights that any improvement must encompass more than the technical (e.g. the adoption of specific techniques and tools), and actively include the social (e.g. addressing employee relationships, raising awareness levels and proactively acknowledging the drivers, and influences, which exist both internal and external to an organisation), since all these aspects are intertwined. Further, this study contributes to the existing STIN literature by showing its suitability for studying security V&V.

Contents

Chapter 1

Introduction

Within this chapter we introduce the research, the overarching research question and the associated research objectives. Further, we justify the organisational and technological focus of the study and provide the supporting definitions necessary for the remainder of the thesis. Specifically, this chapter provides:

Section 1.1: An introduction, providing motivation for the research.

Section 1.2: Justification of the organisational focus.

Section 1.3: Justification of the technological focus.

Section 1.4: The overarching research question, the associated research objectives and resultant contributions.

Section 1.5: The structure of the thesis.

Section 1.6: The work published.

We begin by providing an introduction.

1.1 Research Introduction and Motivation

Within this thesis we examine how security verification and validation is practiced, supported, and perceived, within small to medium-sized software enterprises based within the United Kingdom. Further, we show how the activities associated with this complex socio-technical practice are influenced by the socio-technical characteristics of such organisations - specifically, by determining whether organisational information security culture influences security verification and validation practice and by establishing whether organisations can possess a mature security verification and validation practice without a correspondingly mature software verification and validation practice. This is necessary given the reliance on software for our everyday existence: software is deeply embedded within the fabric of modern society, impacting its social, political, economic, and cultural development [Sjøberg and Grimstad, 2010]. Such reliance requires a dependability not universally evident in today's software. Certainly, there are many known (and probably unknown) instances of software failures and software project failures e.g. [Neumann, 1995, Anderson, 2001, Patton, 2005]. This implies that the producers and consumers of software products are incurring various costs - costs which could potentially be mitigated. Further, although such costs are well known, they are not always easily quantifiable, and can vary in impact from the trivial, to the significant, to the catastrophic. Clearly, the aim of any software producing organisation is to develop (within an acceptable and appropriately bounded resource and time frame) perfectly dependable software that operates deterministically. However, as a human process, software development is prone to fallibility in terms of the interaction between people, and their interaction with specific technical methods and technologies. Therefore, it is necessary to examine both the social and the technical realities surrounding software development.

Developing software, regardless of its intended application (and to some extent its size), is an inherently complex task [Balci et al., 2002, Sawyer, 2004, McGraw, 2006, Sjøberg and Grimstad, 2010]. The situation is further compounded by the many diverse facets of software development and the many socio-technical challenges that can occur throughout the software development lifecycle. Software development is a socio-technical discipline [Sawyer, 2004, Henry, 2005, Sjøberg and Grimstad, 2010, Sedano et al., 2017] and these challenges all conspire to impact and shape the product being developed - sometimes resulting in success, sometimes failure. The facet of software development charged with ensuring a successful product is that of verification and validation (hereafter referred to as "V&V"). Comprising a set of activities, V&V aims to: determine whether a product satisfies its requirements; enable earlier defect

detection and resolution; and assist in gauging product quality attributes. Effectively, providing confidence to an organisation that the product they are developing, and will subsequently be releasing, is fit for purpose. These activities are certainly not immune to the socio-technical challenges generally faced on a software project - both introducing new challenges and heightening others. It is, therefore, essential that such activities charged with ensuring and determining the shape - and the "goodness" of a product - are fully understood in both a social and technical context.

Further restricting our focus, we adopt a specific organisational and technological context. In terms of the former, we observe that many software producing organisations are small to medium-sized enterprises (hereafter referred to as "SMEs"). Whilst such organisations are known for making a significant contribution to a country's economy [Berry, 2007, Mac an Bhaird, 2010], it is not just the numeric quantity and economic contribution of these organisations that warrants focus, but rather that such organisations are developing significant products [Fayad et al., 2000] - products which must also be developed in an efficient, timely, and resource bounded manner, as well as being considered dependable and usable by a consumer. Further, where some socio-technical challenges surrounding software development will be universal (i.e. will be faced in small and large organisations alike), there are clear instances where the socio-technical realities surrounding software development, within a smaller organisation, will differ to those faced within larger organisations e.g. employees wearing multiple hats are more likely to be found within smaller organisations [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018]. In terms of the technological context, adopting a security perspective becomes clearly pertinent when considering the impact to the producer and consumer of a software product which, on being released, is found to contain an exploitable vulnerability. Further, security is an interesting topic of focus from a V&V perspective as it requires the blending of several domains of knowledge and expertise. It also potentially requires two or more functions, within an organisation, to work closely together to achieve the desired outcome of producing secure software i.e. security itself is a socio-technical subject [Coles-Kemp and Hansen, 2017]. This clearly involves the integration of both social and technical characteristics, within an organisational context, that, once again, need to be fully understood and appreciated.

Acknowledging and elaborating upon the above, we enumerate several motivational influences for undertaking such a study:

- Software is ubiquitous and fundamental to modern society: this should be considered a truism given software is now central to modern existence and an integral part of many products and services [Kontio, 1998, Messerschmitt and Szyperski, 2003, Tian, 2005, Whittaker, 2012, van Genuchten and Hatton, 2013, Schleife

et al., 2017, Forsgren and Kersten, 2018]. Effectively, software is everywhere [Messerschmitt and Szyperski, 2003, McGraw, 2006, Rice, 2007, van Genuchten and Hatton, 2013, Sánchez-Gordón et al., 2017]. This increased reliance has resulted in a parallelled increase in the risk that software will either fail or will be used for malicious intent: "[i]f ubiquitous computing means ubiquitous software, then ubiquitous software means ubiquitous insecurity" [Rice and Bucholz, 2007, p. 2538]. To sustain this situation software needs to "provide the required capability, be of sufficient quality, be available when promised, and get delivered at an acceptable price" [Eeles and Cripps, 2010, p. 1]. All of these aspects are influenced by V&V in some form. Further, the majority of software producing organisations are small and are developing significant products [Fayad et al., 2000]. The focus on software SMEs is extremely pertinent when considering the role they play in the global software industry [Gleirscher et al., 2012].

- More than 50% of all released software contains defects that affect its execution in some form [Shull et al., 2002]: as software is constructed by humans, it is not perfect [Patton, 2005] (defects are generally caused by human error [Birolini, 2017]). This emphasises the importance of studying both the social (i.e. human element) and the technical. Defects negatively affect both producers and consumers of software products, see Section 1.3.4.

- The software industry spends more money on locating and addressing defects than any other activity: V&V comprises a substantial share of a project's budget. For example, software testing typically accounts for at least 50% of a project's costs [Harrold, 2000, Myers et al., 2011] (with this number increasing for safety-critical software [Ammann and Offutt, 2008]). More time is also being spent on such activities during software development. This is discussed further in Section 1.3.2, however, it is clearly concerning that considerable financial outlay, and human effort, is being expended on activities tasked with locating defects but yet, as above, we continue to release software with defects - often making the same mistakes [Anderson, 2008]. This indicates a problem with V&V practice, and further, one which cannot be solved purely by technical means.

- The impact of vulnerable software is well known and far reaching in its consequences: information insecurity incurs significant costs (reportedly in the billions [Schneier, 2007]) and, although defects are generally costly to users and organisations [NIST, 2002, Vieira et al., 2006, Everett and McLeod, Jr., 2007], vulnerabilities are more costly in terms of business impact [McGraw, 2006]. If an organisation's products are considered insecure they can go out of business [Wysopal

et al., 2006]. Further, as with making the same mistakes during software development, vulnerabilities are no exception [Piessens, 2002, McGraw, 2008, Anderson, 2008]. These issues are discussed further in Sections 1.3.3 and 1.3.4.

- That there is a distinction between theory and practice: this distinction has long been observed when it comes to V&V [Hamlet, 1988, Bertolino, 2003]. Some of the observations made indicate: difficultly transferring research results to industry; that what has been developed is proving intractable in practical application; and that practitioners are neglecting research (which is compounded by researchers writing for one another and thus losing contact with the realities practitioners face [Parnas and Lawford, 2003]). These observations are discussed further in Section 2.6. However, as the current state of security V&V practice is effectively unknown, this hinders the ability in understanding how to improve upon current practice (and, as above, there is evidently room for improvement). This is discussed further in Section 1.3.6 and has been highlighted as an area warranting focused research within an SME context [Kreeger and Harindranath, 2012].

- Treating V&V solely as a technical concern: it is apparent that the "human element" is often overlooked (aside from education, which is discussed in Section 2.6). Whilst there is some treatment of the social aspects of V&V e.g. [Cohen et al., 2004, Acuña et al., 2006, Devito Da Cunha and Greathead, 2007, Rooksby et al., 2009, Shah and Harrold, 2010, Zhang et al., 2014, Zhang et al., 2018], the majority of the available literature primarily focuses upon the technical e.g. studies comparing specific methods or tools, or the introduction of new techniques, thereby neglecting the social aspects (this is reflected within Chapter 2). As V&V is a socio-technical activity, we justify the importance of adopting a socio-technical perspective within Chapter 3.

The author is also motivated by his own experiences of having previously performed security V&V for an SME; whereupon, it was evident that there was room for improvement in terms of actual practice and that the activities themselves were strongly influenced by both the surrounding social and technical realities i.e. resolving just the technical issues would not necessarily have led to the activities becoming markedly improved within the organisation. Therefore, to improve security V&V practice, we acknowledge that it is essential to:

- Observe the surrounding socio-technical realities of such practice: V&V is a socio-technical discipline. Therefore, separating the social and technical aspects, and focusing on just one, would not present a meaningful or accurate picture.

- Study such practice within an industrial context: studies conducted within an academic environment are unlikely to generalise to an industrial context. To understand industrial practice, it is essential to study such practice within an industrial context.

Therefore, we undertake a mixed methods, sequential explanatory design based study, which focuses on security V&V within UK-based software SMEs. Further, guided by the adoption of the Socio-Technical Interaction Network framework (STIN, see Section 3.3.6), we focus on both the social and technical aspects of security V&V.

Having provided an introduction, we now refine and justify our focus by examining the elected organisational and technological contexts (Sections 1.2 and 1.3 respectively).

1.2 Organisational Focus

Our organisational focus is defined, and justified, within the following sections.

1.2.1 Small to Medium-Sized Enterprises (SMEs)

An enterprise is "any entity engaged in an economic activity, irrespective of its legal form" [European Commission, 2015, p. 9]. In other words, an organisation developing a tangible product intended for the consumption of, or the provision of a service to, a consumer. As our focus concerns understanding the processes and activities involved in creating a product, organisations only providing a service are not considered. However, further restricting our focus, we observe that several different designations can be applied to an organisation to help classify it by size: "micro", "small", "medium" and "large". These designations are typically determined by a combination of headcount, annual turnover and annual balance sheet totals.

Although defining a small business is deemed controversial [Street and Meister, 2004], we adopt the European Commission definition of an SME [European Commission, 2015]. This restricts our focus to organisations employing < 250 people and having either an annual turnover of $\leq$ €50 million or an annual balance sheet total of $\leq$ €43 million (see Appendix A for the value thresholds). We adhere to this definition when determining our sample (Section 5.1.2) and when performing data analysis and presenting our results (Chapters 5, 6, 7 and 8).

1.2.2 Software Enterprises

The Standard Industrial Classification of Economic Activities (SIC) system classifies distinct sectors of industry [Office for National Statistics, 2009]. Utilising this sys-

tem, we focus on organisations falling within class 62.01 ("Computer programming activities"), which encompasses two sub-classes:

- 62.01/1: "Ready-made interactive leisure and entertainment software development".
- 62.01/2: "Business and domestic software development".

This class includes the "writing, modifying, testing and supporting of software" and "designing the structure and content of, and/or writing the computer code necessary to create and implement", for example, system and application software i.e. these are organisations developing software products. Therefore, our organisational focus comprises software SMEs.

1.2.3 Software Products

Although a simplification, software can be classified into the following categories:

- Application: typically providing a specific, well-defined set of functionality to a user, or other application software e.g. a typesetting application.
- System: provides a platform for the stable execution of application software by affording process and memory management i.e. the operating system.
- Middleware: acts as an intermediary, enabling applications, possibly running on different operating systems, to interoperate with one another.

Whilst we do not exclude organisations on the basis of the above categories, we observe that mainstream operating systems are typically either proprietary, and developed by large organisations (e.g. Microsoft Windows and Mac OS), or are open source (e.g. Linux). This would suggest that most software SMEs develop application software. However, this is an ideal combination on which to focus since the majority of vulnerabilities reported are within application software [Han et al., 2009, Kizza, 2017], with the majority of operating systems being considered more secure [Howard and Lipner, 2006].

1.2.4 Justification of Organisational Focus

SMEs are fundamentally important to the majority of the world's economies [Berry, 2007, Mac an Bhaird, 2010]; this is reflected, for example, within Europe [European Commission, 2015], East Asia [Harvie and Lee, 2002] and the UK (comprising 99.9% of

the organisations and accounting for 60% of the employment and 51% of the turnover within the private sector) [Department for Business, Energy & Industrial Strategy, 2017]. Globally, SMEs comprise at least 95% of all registered enterprises [International Finance Corporation - World Bank Group, 2010]. Indeed, 95-99% of organisations, within any country, are SMEs: employing between 40-60% of all workers, and contributing between 40-50% of the Gross Domestic Product (GDP), therefore, they are seen as deserving focused attention [Hax, 2010].

Software, as a valuable commodity, has impacted various industries and will continue to do so [van Genuchten and Hatton, 2013]. Certainly, the software industry has become crucial to the global economy and continues to experience rapid growth [Messerschmitt and Szyperski, 2003]. Therefore, it is understandable that the software industry has become central to the UK economy and economically important in its own right (with clear impact on other UK industry sectors) [Department for Business, Innovation & Skills, 2011]. The UK software industry has not only experienced growth, but is viewed as being both strong and innovative [Microsoft et al., 2008]. Notably, growth of the software industry has been credited for producing small companies [Fayad et al., 2000], with the growing importance of SMEs leading to an increased focus on them [Mac an Bhaird, 2010, Teruel-Carrizosa, 2010]; and when this is coupled with the ubiquity of software itself, and the growth and contribution of the software industry to national economies (see, for example, [Schleife et al., 2017]), it leads to an intersection of areas warranting attention. That software SMEs are known to operate within competitive markets further emphasises this [Feldmann and Pizka, 2003].

However, SMEs are disadvantaged in cultivating secure employee behaviour [Dojkovski et al., 2007], with security practices not being upheld within such organisations [Dimopoulos et al., 2004] (although others have shown signs of improvement e.g. [Department for Business, Innovation & Skills, 2013]). A lack of resource, staff expertise, management support and budget are some of the reasons cited for information security challenges within SMEs [Dojkovski et al., 2006, Dojkovski et al., 2007, Williams, 2009, Dojkovski et al., 2010]. Notably, many studies continue to show that the attitudes of people, as well as their lack of security awareness, are the most significant causes of security incidents [Furnell, 2007]. Information security is a socio-technical subject, with humans interacting with information and information systems [Ngo et al., 2009, Okere et al., 2012, Coles-Kemp and Hansen, 2017]. Unsurprisingly, humans are a significant factor in information security [Williams, 2009] and it was this acknowledgement which provoked a paradigm shift from a technical to a socio-cultural approach in terms of its study [Connolly and Lang, 2012] (which further reinforces the adoption of STIN, see Sections 3.3.6 and 3.4).

Having justified the organisational context, we reflect upon the importance of software SMEs, not only in terms of developing products as efficiently as possible, but also in satisfying the quality expectations of the intended consumers. V&V comprises a set of activities which can assist organisations in meeting these objectives. This forms the technological context.

1.3 Technological Focus

Within this section we examine the process of V&V and the associated costs.

1.3.1 Software Verification and Validation (V&V)

Software V&V has been a topic of focus since the early days of software development. Examining each aspect in turn:

- Verification: is "the process of evaluating a software system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase" (and is typically associated with review-based activities) [Burnstein, 2003, p. 6].
- Validation: is "the process of evaluating a software system or component during, or at the end of, the development cycle in order to determine whether it satisfies specified requirements" (and is typically associated with test-based activities) [Burnstein, 2003, p. 6].

The activities associated with software V&V can be applied at different stages during software development, with their involvement dependent on the software development methodology adopted. For example, within the Waterfall model it is a distinct phase starting later on in development; within the V-Model there are integrated software V&V phases corresponding to individual development phases. Although the terms "verification" and "validation" have distinct meanings, this is not always evident [FDA, 2002, Dybkaer, 2011]. Verification asks: "are we building the product right?" (ensuring that the product conforms to its specification), whereas validation asks: "are we building the right product?" (ensuring that user needs are met). However, like others (e.g. [Burnstein, 2003, Tian, 2005, Xiong, 2011]), we view test and review-based activities as spanning both the verification and validation domains.

We now examine the activities associated with software V&V, as well as what they aim to find, namely, defects.

1.3.1.1 Software Testing

Software testing is the process of analysing executable software. It is one of V&Vs most important and commonly used methods [Vieira et al., 2006, Ramler et al., 2006, Dasso and Funes, 2007, Fernández-Sanz et al., 2009] and has several, general objectives, for example: validating that the product satisfies its requirements, enabling earlier defect detection and resolution and characterising system performance at an earlier stage [IEEE, 2008, Xiong, 2011]. It can also be used to gauge product quality attributes, such as: security, usability and reliability [Burnstein, 2003, Bertolino, 2003].

Software testing encompasses a range of activities (from test planning and test design, through to test implementation and test execution, as well as reporting and management) and, whilst considered the most common aspect of software V&V, it is generally accepted that it should be complemented by software reviews [Wood et al., 1997, Futrell et al., 2002].

1.3.1.2 Software Design and Code Reviews

Software inspections, walkthroughs and reviews (hereafter collectively referred to as "software reviews") are an important means of producing high-quality software [Aurum et al., 2002, Berling and Runeson, 2003, Nelson and Schumann, 2004, Kollanus and Koskinen, 2006, Uwano et al., 2008, Kollanus, 2009, Humayun et al., 2010, Shull et al., 2012]. Besides software testing, they are the most common software V&V activity [Tian, 2005], providing a means of detecting "errors, violations of development standards, and other problems" [IEEE, 1990, p. 40]. Further, they can be applied to artefacts other than those requiring execution (i.e. code), for example: requirement specifications, designs, test designs and user documentation. Therefore, whilst we adopt the umbrella term "software review" (like [Rombach et al., 2008]), we distinguish focus in terms of the artefact being reviewed, namely, a product's design or code.

We now define what is meant by a software defect.

1.3.1.3 Software Defects

Although there is no commonly accepted definition, a defect can be viewed as a "deviation from quality" [Mäntylä and Lassenius, 2009, p. 431]. This includes the non-fulfilment of user requirements, to something interfering with quality improvement [Kelly and Shepard, 2002]. However, typically, a defect is a problem with a product's internal characteristics or external behaviour [Tian, 2005]. It can also be a discrepancy between specification and implementation, or a product which "is difficult to understand, hard to use, slow, or in the software tester's eyes will be viewed by the end user

as just plain not right" [Patton, 2005, p. 15]. Defects typically become an issue when there is insufficient V&V on a project (defect impact is discussed within Section 1.3.4). This can be attributed to, for example, a software development process failing to afford importance to software testing [Vieira et al., 2006]. However, the application of V&V itself is not without cost.

1.3.2 Costs Associated with V&V

The size and complexity of modern software is well known [Balci et al., 2002, McGraw, 2006, Sjøberg and Grimstad, 2010, van Genuchten and Hatton, 2013], with this resulting in a paralleled increase in the complexity of the V&V process [Kreeger and Lativy, 2008]. Such complexity hinders the ability in enumerating the various states of a piece of software, leading to unreliability and a variety of technical and managerial problems to contend with [Brooks, Jr., 1987]. This not only increases the costs associated with V&V (e.g. with more time being spent on software testing due to the diversity and complexity of software [Sung and Paynter, 2006]), but also the number of defects [Xiao et al., 2007]. The relationship between complexity and defects is well known [Banker et al., 1989]; certainly, as software increases in size, it is natural to think that it will contain more defects [Porter et al., 1998]. Therefore, a development process is required that employs quality control at each phase of the process [Vieira et al., 2006]. However, costs arise from such complex human and technology-based processes [Sjøberg and Grimstad, 2010].

For example, software testing is a costly activity [Rankin, 2002, Xiao et al., 2007, Larusdottir et al., 2010] - possibly the most expensive [Kasurinen et al., 2010] - commonly accounting for at least 50% of a project's cost [Harrold, 2000, Memon, 2002, Myers et al., 2011] (with this number increasing if the software is of a safety-critical nature [Ammann and Offutt, 2008]). It also accounts for approximately 50% of the elapsed time of a project [Myers et al., 2011]. Significantly, the amount of testing being performed is increasing due to the demand for higher quality products [Taipale et al., 2005]. However, overzealous testing can result in a product that is delayed to market and overpriced [Dalal, 2003] (knowing when to stop testing is a key question [Voas and Miller, 2012]). Similarly with software reviews, whilst there are many different associated costs and benefits [Porter et al., 1998], they are considered expensive, labour-intensive and time-consuming [Kelly and Shepard, 2002, Parnas and Lawford, 2003]. This is in part due to review teams typically comprising between 4-5 people [Aurum et al., 2002]; often relying on senior engineers [Nelson and Schumann, 2004]; and regularly involving meetings [IEEE, 1990, Votta, Jr., 1993, Johnson and Tjahjono, 1998]. Software design and code review activities account for approximately 15% of a project's cost [Fagan,

1986].

V&V can also incur a variety of other costs, for example, the preparation and development of the test framework and test environment; hardware and software acquisition; and training [Burnstein, 2003]. Notably, test automation is seen as being both cost-effective and expensive [Everett and McLeod, Jr., 2007] and, aside from the initial cost of acquiring tools, their uptake, and successful deployment, can introduce additional costs in terms of training and organisational support. In summary, V&V comprises a significant share of a project's budget [Andersson and Runeson, 2002]. Unfortunately, the existing literature does not provide much elaboration on these costs within an SME context - either in terms of software V&V or security V&V. It is within Section 1.3.4 that we examine the costs incurred by an organisation when failing to adequately perform these activities.

We now examine software security and vulnerabilities.

1.3.3 Software Security and Vulnerabilities

Software security has traditionally been defined by three terms: confidentiality (preventing unauthorised disclosure of information), integrity (preventing unauthorised modification of information) and availability (preventing unauthorised withholding of information or resources) [Fischer-Hübner, 2001, Gollmann, 2011, Pfleeger and Pfleeger, 2012, Stallings, 2017]. These, and other security objectives (such as authentication, authorisation and non-repudiation [Menezes et al., 1996]) when satisfied, collectively help ensure that software functions correctly e.g. maintains data confidentiality and ensures data integrity [Potter and McGraw, 2004]. Therefore, whilst McGraw views software security as the building of software that will continue to function as expected when under malicious attack [McGraw, 2004], we feel that Anderson's view - building software that is dependable in the face of malice or misfortune [Anderson, 2008] - is more encompassing (since vulnerabilities may not just exhibit themselves when software is under deliberate attack).

Therefore, with the understanding that software security preserves the various security objectives enumerated above, we broadly define (based on [McGraw, 2006, Ozment, 2007, Takanen et al., 2008]) a vulnerability as a defect, or weakness, in a software's design, implementation, or configuration, which can be utilised to cause a security breach, or violation of an implicit or explicit security policy. There are many distinct software weaknesses which can lead to vulnerabilities ([The MITRE Corporation, 2020] offers a catalogue of software weakness types covering, for example, SQL injection and the use of hard-coded credentials). However, perhaps more alarming is not the number of different types of vulnerabilities, but that they are common, increasing

in number [McGraw, 2006], and that the same mistakes continue to be made [Piessens, 2002, McGraw, 2008, Anderson, 2008]. This is certainly evident when examining the available statistical data (e.g. [NIST, 2020]) and when acknowledging that many security test vendors have built a business upon this trend [Thompson, 2003] - clearly, we should learn from our mistakes (which is one of the simplest ways we can improve security [Pfleeger and Pfleeger, 2012]). However, discovering vulnerabilities is hard, with developers lacking security expertise further compounding the situation [Austin and Williams, 2011]. The need for test engineers to possess security knowledge has also been identified [Kreeger, 2009, Felderer et al., 2011]. Additionally, and further complicating matters, many traditional test techniques are unsuited to locating vulnerabilities [Thompson et al., 2002, Jourdan, 2009]. In conclusion, even though software security has become a noted concern [Cooper et al., 2009], with increased focused attention, the number of vulnerabilities being reported continues to increase [Jourdan, 2009].

We now examine the impact defects and vulnerabilities have on software producing organisations.

1.3.4 Impact of Defects and Vulnerabilities

The software industry spends more money on locating and addressing defects than any other activity [Jones, 2012]. Further, it is reported that more than 50% of all released software contains defects that affect its execution in some form [Shull et al., 2002] (an industry standard suggests 25% of a product's defects are found by developers, 50% by test engineers, with the remainder by end users [Ahmed, 2009]). The annual cost of defects - to both users and organisations - is in the billions of dollars [NIST, 2002, Vieira et al., 2006, Everett and McLeod, Jr., 2007], with Schneier explicitly indicating that "[i]nformation insecurity is costing us billions" [Schneier, 2007, p. 3]. This is clearly an unsustainable situation.

It has long been acknowledged that the costs of addressing defects increase as a project nears completion [Boehm, 1981, Adrion et al., 1982], therefore, the earlier a defect is identified, the less costs incurred by the software producing organisation [Ammann and Offutt, 2008]. This is further reinforced when assigning monetary values to the cost of addressing defects, as they become identified, during the various stages in the software development lifecycle e.g. [The Economist, 2003, Jones, 2012]. Such examples exemplify the 1:10:100 rule, showing that it is more economical to secure software during its design [Boehm and Basili, 2001, Shull et al., 2002, Geer, Jr. et al., 2003, Gregory, 2015] i.e. vulnerabilities become more expensive to fix [McGraw, 2008]. However, whilst many statements, based on the cost per defect metric, hide several problems (for example, the value of improved software quality is not captured i.e. higher

quality results in decreased development times and expenditure) [Jones, 2012], the examples above clearly identify, and reinforce, that addressing defects after release is significantly more expensive. This is also true of vulnerabilities [Austin and Williams, 2011, Ransome and Misra, 2014], which have a higher associated business impact cost than defects [McGraw, 2006].

To fully appreciate the impact of vulnerabilities, we observe that some of their costs, and impact, can be difficult to quantify in monetary terms, and go beyond the immediate cost of defect remediation. For example, damage to corporate reputation (releasing a vulnerable product can generate a public relations nightmare [Coffey and Viega, 2007]) can negatively impact consumer purchasing habits, as well as the ability for the organisation to charge a premium for their products. Security concerns can also result in a lack of consumer trust [Centeno, 2003, Rainys, 2006, Coffey and Viega, 2007, Halaweh and Fidler, 2008] which is inherently entangled with customer loyalty [Flavián and Guinalíu, 2006]. Notably, [Telang and Wattal, 2007] found that the announcement of a vulnerability correlated with a loss in an organisation's market value on the day of the announcement; as well as losses in market share if the market was competitive or the organisation small. Software SMEs are known to operate within very competitive markets [Feldmann and Pizka, 2003].

Significantly, these are all organisational characteristics that require both time and investment to develop and sustain, and which would be costly to rebuild and re-establish (if possible to do so). Additionally, members of an organisation's engineering team may need to be redeployed to promptly address the reported issues. This would impact other projects by reducing the available engineering resource from working on the next release [Hartman, 2002]. It would also delay time to market, an important factor for software organisations [Conradi and Fuggetta, 2002]. In turn, this would further compound the fact that "many software markets have become fiercely competitive and have demanded faster software product delivery" [Fayad et al., 2000, p. 116]. Notably, when products are late to market, or are of poor quality, profitability is reduced [Trew, 2007]. It is, therefore, evident, that vulnerabilities are costly to software producing organisations.

We now examine the V&V activities aimed at identifying vulnerabilities.

1.3.5 Security Verification and Validation (V&V)

Consulting several "standard" V&V texts, we find that little treatment is afforded to security V&V. For example, only dedicating a few pages to security testing and primarily considering it part of system test e.g. [Burnstein, 2003, Myers, 2004]; or, defining it as "the process of attempting to devise test cases that subvert the program's security checks" [Myers et al., 2011, p. 125]. We regard this system test and security

mechanism focus as being rather limited. However, others are starting to cover aspects of security V&V in more depth e.g. [Patton, 2005, McGraw, 2006], with more dedicated texts appearing e.g. [Whittaker and Thompson, 2004, Wysopal et al., 2006, Takanen et al., 2008] (although, within Section 2.6, we observe the distinction between theory and practice i.e. what is stated within textbooks and what is practiced within industry).

Security V&V encompasses the following objectives:

- Demonstrating that the security properties, and the behaviour of the software, remain satisfied, predictable, and secure, even when in the presence of a malicious party.
- Uncovering the presence of vulnerabilities. Thereby reducing the probability of their being found by a consumer.

Notably, a variety of security V&V activities can be applied to meet these objectives and are covered within several secure software development processes (e.g. Touchpoints [McGraw, 2006] and Microsoft's SDL [Microsoft, 2020]) and software security maturity models (e.g. BSIMM [McGraw et al., 2016] and SAMM [OWASP, 2017]). Each of these offers a variety of practices, and guidance, in terms of developing secure software which, collectively, helps emphasise that security V&V can occur throughout software development i.e. there is no reason to focus such efforts solely at the system test level or upon a product's security mechanisms (as might be inferred from several key V&V texts cited earlier).

We now examine the activities which comprise security V&V.

1.3.5.1 Security Testing

Security testing aims to determine whether the security properties referenced within Section 1.3.3 are satisfied [Felderer et al., 2011]. Although very similar to other types of testing (e.g. resources need to be identified, test specifications prepared, tests executed and results recorded), a significant difference between software testing and security testing is that the latter assumes an opponent actively seeking vulnerabilities [Zuccato and Kögler, 2009] i.e. it emphasises what a product should not do [Felderer et al., 2011]. Whilst functional security testing ensures that the security functionality of a product is correctly implemented, this is largely an exercise in verifying that the product's requirements are satisfied. As observed: "software security is not security software" i.e. vulnerabilities can exist either within the security mechanisms themselves, or elsewhere within a product [Potter and McGraw, 2004, p. 82]. Potter and McGraw state that many developers and test engineers mistakenly believe that security only

encompasses a product's security features. This is clearly a limited view, and leads to the mistaken beliefs that only the software which performs a security function needs to be secure, and that the implementation of security functionality makes software secure [Goertzel et al., 2007]. This is why functional security testing should be complemented by risk-based security testing.

Such testing involves identifying the risks (which are grounded in the system's architectural reality and an attacker's mindset) and then devising tests around these risks [Potter and McGraw, 2004]. Therefore, we view risk-based security testing as simulating an attacker's approach, but encompassing a greater focus than just a product's underlying security mechanisms. However, such an approach requires an experienced engineer, "traditional" test engineers will find the adoption of a risk-based approach more difficult than a functional approach (which, as noted above, can probably be addressed via standard functional testing) [Potter and McGraw, 2004]. Potter and McGraw view this problem as one of expertise, and observe that there is then more reliance placed on an engineer's expertise and experience than is desirable (this is echoed by [Michael et al., 2013]).

1.3.5.2 Penetration Testing

Thompson defines penetration testing as: "software testing that's specifically designed to hunt down security vulnerabilities" [Thompson, 2005, p. 66]. Whilst a rather generic interpretation, [McGraw, 2006] acknowledges that the significant differences between penetration testing and security testing lie in the level of approach adopted and when the activity is applied. Specifically, McGraw sees penetration testing as being applied on code completion and when the software is situated within an operational environment. It also encourages the adoption of an outside in perspective [van Wyk, 2013]. Although penetration testing is viewed as being the most frequently applied of security practices, it can also be viewed as being "too little, too late" - which is typically the case when security is not considered an emergent property [Arkin et al., 2005, p. 84]. However, the main value derived from penetration testing is the identification of environmental and configuration problems which, in a financial sense, can still plausibly be addressed towards the end of software development.

1.3.5.3 Security Design and Code Reviews

As with software design and code reviews, security-focused reviews can be performed throughout the software development lifecycle, however, security design reviews should be performed early to help influence a product's design [Meier, 2006]. Further, security-

focused reviews should not only be performed early, but also multiple times [Jaatun et al., 2011]. Whilst software design reviews, in general, can be effective, [Meier, 2006] observes that they do not produce the same results as security-focused design reviews. Meier also highlights the importance of conducting security reviews as distinct reviews in their own right, since: when "you're focused on security, you aren't randomly focused on other quality attributes, except when there's a direct intersection" [Meier, 2006, p. 21]. Although this reinforces the view that security is a quality attribute (as it is viewed by others e.g. [Burnstein, 2003]), by considering security in isolation, further helps emphasise its importance.

In addition, although security code reviews are considered key to building secure software [Lipner, 2000], McGraw observes that those conducting the reviews "regularly suffer from the "get done, go home" phenomenon", thereby leading to a strong start, but a less positive finish [McGraw, 2008, p. 111]. Code reviews can suffer due to their inherent complexity and time-consuming nature [Edmundson et al., 2013], but this can be alleviated through automated source code analysis (removing some of the burden of a manual security code review [McGraw, 2008]), however, [Meier, 2006] cautions that the advantage of conducting a manual review is through the application of contextual analysis.

1.3.5.4 Summary

Whilst many tools and techniques have been devised to help find vulnerabilities, no single tool or technique is sufficient in locating all vulnerabilities or even all types of vulnerability [Austin and Williams, 2011] (Austin and Williams also note that a lack of empirical evidence leaves developers "to decide how they can best discover vulnerabilities on their own"). In summary, we observe that developing secure software requires changes in the way organisations develop software [McGraw, 2006]. McGraw notes that such changes "do not need to be fundamental, earth shattering, or cost prohibitive". This implies that such improvements are within reach of SMEs. Based on the nature and impact of vulnerabilities (see Sections 1.3.3 and 1.3.4 respectively), it is clearly vital that the activities associated with security V&V are adopted throughout the software development lifecycle. It is also apparent that the security V&V activities themselves are not immune to the social and technical influences which impact V&V more generally (see Section 3.2). As [Lipner, 2000] indicates, those performing security-focused test and review-based activities must have the motivation, and the necessary resources available, in order to perform the activities - which further supports the adoption of STIN (which explicitly focuses on motivation and resource flows, see Section 3.3.6).

1.3.6 Justification of Technological Focus

It is generally acknowledged that software is essential to the modern information society [Sjøberg and Grimstad, 2010]. Therefore, it is unsurprising that software is considered a valuable product [Ciolkowski et al., 2002]. However, there are many examples of "software quality lapses that are shaking the public's confidence that software can be used to build safe, secure systems" [Parnas and Lawford, 2003, p. 20]. It is essential that software performs as expected, even when subjected to malicious treatment, as the consequences can be severe [Kreeger, 2009]. The literature is littered with examples of software failures e.g. [Neumann, 1995, Anderson, 2001, Patton, 2005] - thus: "software practitioners and software researchers must improve software's reputation; the only way to do that is to improve software quality" [Parnas and Lawford, 2003, p. 20]. However, although there are calls for high-quality software [Sjøberg and Grimstad, 2010] - with consumer demand increasing interest in software testing and raising the importance of the activity itself [Machado et al., 2010, Ahmed, 2011] - producing it is perceived as a considerable challenge [Rodrigues et al., 2010]. Within very small software organisations "even a small problem [...] can have huge repercussions" [Basri and O'Connor, 2010, p. 153] (which is certainly the case with vulnerabilities, see Section 1.3.4). In summary, software quality is important for the continued growth of software organisations, regardless of their size and type of product [Basri and O'Connor, 2010]. However, we recall that the majority of software producing organisations are small and are developing significant products [Fayad et al., 2000]. The author's experience of working for a software SME confirms this.

The activities associated with V&V are closely coupled to software quality [Adrion et al., 1982], with software testing recognised as one of the most important means of ensuring software quality [Vieira et al., 2006, Takanen et al., 2008]. However, software testing cannot be considered trivial due to its many challenges [Rodrigues et al., 2010] (both researchers and practitioners view software testing, within small organisations, as an important area, but one also facing many challenges [Pyhäjärvi et al., 2003]). Notably, whilst software testing was considered "difficult, time consuming, and often inadequate" [Adrion et al., 1982, p. 161], this is still the case [Whittaker, 2000, Wiederseiner et al., 2010, Myers et al., 2011]. Software review-based activities have also been viewed as complex and time-consuming [McGraw, 2008, Edmundson et al., 2013]. Additionally, [Patton, 2005], who details several well-known software failures, observes that as software is created by people, it is not perfect. Collectively, these further strengthen the need to examine security V&V from a socio-technical perspective (see Chapter 3).

However, although the importance of V&V has been recognised within SMEs and large organisations alike, there is an identified need, within smaller organisations, to

understand and improve aspects of V&V e.g. software testing [Pyhäjärvi et al., 2003]. Notably, SMEs are regarded as having a rather limited view in terms of the techniques to adopt and apply [Tervonen and Harjumaa, 2004]. Further, although there exists a plethora of research on V&V, it has had a limited impact on practice [Parnas and Lawford, 2003, Lethbridge et al., 2007]. Additionally, there is also limited treatment, within the existing literature, as to how security V&V is performed within software SMEs (as evidenced by Chapter 2 and summarised in Appendix C). It is thus apparent that security V&V practice, within UK-based software SMEs, is an empirical unknown [Kreeger and Harindranath, 2012]. Therefore, it is essential that we address this unknown, especially when acknowledging the ubiquitous nature of software, its import (governments, industry and academia all acknowledge security as a major concern [Cooper et al., 2009]), as well as the reality that software is frequently impacted by vulnerabilities [Eschelbeck, 2005].

In conclusion, practice has to be understood for the benefit of both industry and the research community and, although stated in the context of regression testing: "[t]here is a need for researchers to better understand the needs and practices in industry" [Engström and Runeson, 2010, p. 3]. We observe that regression testing is a fundamental and long standing topic, however, with the impact of vulnerabilities, and the importance of security V&V, it is clearly concerning that we have so little visibility on actual security V&V practice. Empirical studies, based within industry, are important for advancing knowledge in areas such as software testing [Gittens et al., 2002]. Our research aims to illuminate the area of security V&V, as practiced within UK-based software SMEs, for the benefit of both industry and the research community.

1.4 Research Question, Objectives and Contributions

Within this section we state the overarching research question, the associated research objectives and the contributions made by this thesis.

1.4.1 Research Question

The research question (RQ) addressed by this thesis is:

RQ: *Within UK-based software SMEs, how is security V&V performed, and how, as a process, is it influenced by the socio-technical characteristics of such organisations?*

This is, in turn, comprised of three research objectives which further reflect, and restrict, the scope of how the research question is addressed.

1.4.2 Research Objectives

We decompose the research question into three, specific, research objectives (RO):

RO1: *To examine how security V&V is practiced, supported, and perceived, within UK-based software SMEs*

There is an identified need for empirical software engineering studies [Wood et al., 1997, Sjøberg and Grimstad, 2010], especially concerning V&V practice, within an industrial context [Torkar and Mankefors, 2003, Berling and Runeson, 2003, Ellims et al., 2004, Kollanus and Koskinen, 2006, Itkonen et al., 2009, Larusdottir et al., 2010, Engström and Runeson, 2010, Gren and Antinyan, 2017]. However, on examining many of these previous studies, we find a distinct lack of focus on security V&V practice. Empirically, our knowledge of the situation within SMEs is even more bleak. It is only by examining current practice that we can begin to understand how to improve upon such practice. Further, we highlight the need to examine both the social and technical aspects of security V&V practice since, focusing on just the technical aspects, for example, would only present a partial picture, which would not be meaningful in terms of improving practice. Therefore, as surveys are useful for capturing the current status of a situation [Wohlin et al., 2003], and given both their prevalence and suitability for studying software engineering processes and practices more generally (e.g. [Singer and Vinson, 2002, Punter et al., 2003, Cater-Steel et al., 2005, Smith et al., 2013]), we begin by employing a survey to examine security V&V practice. Additionally, and acknowledging the utility of acquiring both quantitative and qualitative data when undertaking an empirical software engineering study [Seaman, 1999], we conduct a series of follow-up interviews (as with surveys, interviews are widely employed within empirical software engineering research [Hove and Anda, 2005]). Notably, such an approach is particularly desirable when researching an underexplored area [Mandić et al., 2009, Venkatesh et al., 2016]. We provide further justification for the adoption of a sequential mixed methods approach within Chapter 4.

RO2: *To determine whether organisational information security culture influences security V&V practice*

Individuals, situated within any type of organisation, are influenced by various organisational cultures [Kuusisto and Ilvonen, 2003]. This includes an organisation's information security culture, which encompasses "all socio-cultural measures that support technical security methods" [Schlienger and Teufel, 2003, p. 46]. However, although it is apparent that various definitions of such a culture exist [Williams, 2009], it is also

apparent that much of the work on forming an information security culture, within an organisation (including SMEs), centres on the organisation itself, and not, necessarily, on the products being developed by the organisation. Further, we note that the purpose of an information security culture is for information security to become a natural aspect to an employees' daily activities [Sánchez et al., 2010]; we, therefore, intend to determine whether an organisation's information security culture influences the level of security V&V performed. Given that surveys can be used to perform organisational assessments [Singh et al., 2009], and that they have been employed extensively within many existing information security culture studies (see Section 4.5.3), we also employ a survey. Notably, we consider the anonymity afforded by survey adoption as being essential for when studying a highly sensitive subject (see Section 5.2). This is also applicable to the first and third research objectives (in terms of understanding security V&V practice) and, as with those objectives, we also perform follow-up interviews to further explore, and corroborate, the themes identified within the initial survey.

RO3: *To establish whether organisations can possess a mature security V&V practice without a correspondingly mature software V&V practice*

It is acknowledged that an increase in organisational maturity, in terms of software development, leads to software processes becoming more defined, managed, and measured, in turn, leading to organisations having improved control over the development process itself [Bashir and Goel, 1999, Burnstein, 2003]. This, in the case of software testing for example, can lead to an improvement in software product quality [Burnstein, 2003]. Certainly, the quality of a software product is related to the quality of an organisation's software processes [Burge et al., 2008]. However, much of the existing literature regarding software V&V maturity has been rather generic i.e. security V&V has received little attention. It is through establishing the maturity levels of both software and security V&V practice, within UK-based software SMEs, that we can then understand their relationship (for example, establishing whether their respective maturity levels are synchronised, or whether an organisation can possess a more mature security V&V practice). Therefore, and given their shared heritage, as well as the need to contrast both software and security V&V practice - and their associated maturity levels - we employ the same set of questions when addressing the first and third research objectives (see Section D.1).

1.4.3 Specific Research Contributions

By addressing the research question, and the associated research objectives, we claim the following contributions:

- We provide a holistic, socio-technical account of how security V&V is practiced, supported, and perceived, within UK-based software SMEs. Such practice was, hitherto, an empirical unknown.
- By deriving a similar understanding, in terms of software V&V, we are able to establish the relationship between an organisation's software and security V&V practices. We find no evidence, despite their shared heritage, to suggest that an organisation can possess a mature security V&V practice without a mature software V&V practice.
- By obtaining an understanding of information security culture, within UK-based software SMEs, we are able to determine how this organisational characteristic influences security V&V. Specifically, we find that an information security culture does influence security V&V practice.
- Whilst STIN has been employed within a variety of contexts, including software engineering, we show its suitability when applied to the study of V&V. This, to our knowledge, is the first time STIN has been applied to the study of V&V.
- We demonstrate the value of employing a sequential, explanatory mixed methods design, when conducting a STIN-framed, empirical software engineering study.

These are discussed within Section 8.2.

1.5 Thesis Structure

The thesis takes the following structure:

- Chapter 1: we introduce the research, the overarching research question and the associated research objectives. Further, we justify the organisational and technological focus of the study and provide the supporting definitions necessary for the remainder of the thesis.
- Chapter 2: we perform an analysis of the relevant literature to demonstrate the validity and value of the proposed research.
- Chapter 3: we show that software engineering, and V&V, are socio-technical activities. We then justify the adopted socio-technical framework.
- Chapter 4: we justify the approach, and the research methods adopted, to address the research question and the associated research objectives. We then detail how the survey instrument and the interview guide were developed.

- Chapter 5: we present the first phase of data collection. This includes detailing the pre-testing performed, the sampling strategy, the method of instrument distribution and discussion of the response rate. We then analyse the data received and reflect upon the findings and the approach taken.
- Chapter 6: we present the second phase of data collection; whereupon, we analyse the data received and reflect upon the findings and the approach taken.
- Chapter 7: we discuss, and bring together, the findings obtained during the two phases of data collection and analysis performed.
- Chapter 8: we summarise the thesis contributions and the implications for theory and practice. We then examine and discuss potential limitations and future research directions.

1.6 Publications

The following publications are a result of the research performed:

1. **Kreeger, M. N.** and Harindranath, G. (2012). Security Verification and Validation by Software SMEs: Theory versus Practice. In: *Proceedings of the 2012 International Conference on Information Resources Management (Conf-IRM 2012)*, Vienna University of Economics and Business.
    - Based on material from Chapters 1 and 2, provides justification for the thesis.
2. **Kreeger, M. N.** and Harindranath, G. (2017). Security V&V Within Software SMEs: A Socio-Technical Interaction Network Analysis. In: *Proceedings of the 2017 International Conference on Information Resources Management (Conf-IRM 2017)*, University of Chile.
    - Based on material from Chapters 1, 3, 4 and 5, justifies the adoption of STIN for studying V&V and presents the findings from the first phase of data collection and analysis.
    - Awarded Best Conference Paper Runner Up.

Chapter 2

Literature Review and Background Analysis

Within this chapter we perform an analysis of the relevant literature to demonstrate the validity and value of the proposed research. Specifically, this chapter provides:

Section 2.1: An introduction, detailing how the literature review was approached and structured.

Section 2.2: A review of the literature concerning the test-based activities.

Section 2.3: An examination of the literature concerning the review-based activities.

Section 2.4: Tool use and framework adoption is examined.

Section 2.5: Organisational maturity, processes and culture are examined.

Section 2.6: The distinction between V&V theory and practice is assessed.

Section 2.7: Our findings, following engagement with the existing literature, are summarised.

We begin by detailing how the literature review was approached and structured.

2.1 Introduction

As a research area increases in maturity, it becomes necessary to summarise the associated growing body of literature [Brereton et al., 2007, Petersen et al., 2008]. Specifically, it is by reviewing the existing literature that we are able to identify where gaps exist and are thus, in turn, able to show where our research sits within this body of literature [Budgen and Brereton, 2006, Dybå et al., 2007]. As such, we begin by first defining our search strategy, our inclusion and exclusion criteria and how we structured our literature review and background analysis.

Given the technological focus of this thesis (see Section 1.3), we elected to perform a series of initial, focused searches within the following digital libraries: ACM Digital Library [ACM, 2020], IEEE Xplore Digital Library [IEEE, 2020] and Springer Link [Springer, 2020]. Our initial focus on these digital libraries is supported by others. For example, [da Mota Silveira Neto et al., 2011, do Carmo Machado et al., 2014] state that, together, these cover many of the leading software engineering publications and are thus, in turn, likely to include the major test-related research available. Similarly, when examining test-driven development, [Kollanus, 2010] elected to use the ACM, IEEE and Springer digital libraries as they covered most of the relevant, peer-reviewed publications (this is echoed by [Mäntylä et al., 2015] from a software test and release perspective). These digital libraries have also been heavily relied upon when undertaking research into software processes more generally e.g. [Martínez-Ruiz et al., 2012]. Thus, collectively, they offer access to some of the most relevant publications, such as: *Communications of the ACM*, *Empirical Software Engineering*, *Empirical Software Engineering and Measurement* (ESEM), *IEEE Software* and the *International Conference on Software Engineering* (ICSE). Notably, many of these journals and conferences sit amongst the premier outlets for software engineering and empirical software engineering research [Sjøberg et al., 2005].

Having identified the primary data sources, we then performed a series of searches across these digital libraries. The search terms employed, and the number of papers returned before applying the inclusion and exclusion criteria, are detailed within Appendix B. However, it is apparent, even when employing relatively simple search strategies, that the number of results returned is limited e.g. searching for *"security testing" AND "empirical"* through IEEE results in 28 papers being returned (similarly, *"security testing" AND "survey"* returns 13 papers and *"security testing" AND "interviews"* returns 4). Notably, these results are echoed when substituting *"security testing"* with *"penetration testing"* and when using the ACM digital library. They are also reflective of the initial, quantitative focus of empirical software engineering stud-

ies more generally [Basili, 2006]. In contrast, employing the search terms *"software testing" AND "empirical"* returns many hundreds of papers. Further, when examining the papers returned, it was often necessary to exclude them (see Appendix B for the inclusion and exclusion criteria applied). For example, of the 13 papers returned from the *"security testing" AND "empirical"* search through ACM, only one had some demonstrable relevance, with the others solely focused on the technical e.g. [Yang et al., 2016, Pashchenko et al., 2017, Parizi et al., 2018]. Similar is found when searching ACM for *"security testing" AND "survey"* e.g. [Zangooei et al., 2012, Su et al., 2017]. These examples continue to support the view that the existing V&V literature is too focused on the technical [Rooksby et al., 2009].

Others also report encountering "a low signal to noise ratio" when performing a systematic search of the empirical software engineering literature [Mäntylä et al., 2015, p. 1413]. Specifically, Mäntylä et al. highlight that whilst many of the papers returned matched their search terms, they lacked empirical data appropriate to their specific contexts. Therefore, like Mäntylä et al., due to the limited numbers of papers being identified, we elected to also use Google Scholar [Google, 2020] and, in our case, the university's LibrarySearch [RHUL, 2020] as well. Both of these encompass multiple digital libraries and databases (including the ACM, IEEE and Springer digital libraries) and, therefore, dramatically broadened the search scope. However, whilst the use of Google Scholar and a university's online search facility have been adopted, and recommended, by others (e.g. [Kitchenham and Charters, 2007, Brereton et al., 2007, Wohlin, 2014, Shafique and Labiche, 2015, Landman et al., 2017]) we found, given the increased set of papers being returned, that it was necessary to employ a less systematic approach to paper discovery. Specifically, it became necessary to both broaden and tighten our search terms, whilst continuing to adhere to our defined inclusion and exclusion criteria. It also became necessary to employ a degree of snowballing, as appropriate, in order to discover other, relevant literature. Ultimately, the need to adopt such an approach helps confirm that security V&V is an empirical unknown (see Section 1.3.6 and [Kreeger and Harindranath, 2012]).

However, with a set of papers identified, it was then necessary to decide how to group and structure the literature discovered. As captured within Sections 1.3.1 and 1.3.5, both software and security V&V involve a series of activities. Broadly, these activities can be classified as falling into one of two distinct sets of activities: test-based activities and review-based activities [Burnstein, 2003, Tian, 2005]. As such, we elect to group the identified literature into these two themes (see Sections 2.2 and 2.3). Notably, and given that the majority of the existing literature typically only addresses one of these themes at a time, we consider this structure appropriate. In terms of

identifying the activities which fall within these two themes, this was determined by examining a series of software security maturity models and software V&V maturity models (these are discussed and summarised within Appendix M). Therefore, and given the focus on software and security V&V, we ensure that both software and security-focused activities are covered within Sections 2.2 and 2.3 of the literature review and background analysis.

Additionally, as a socio-technical discipline, we acknowledge that software engineering involves a combination of both human and non-human interactors (this is discussed further within Chapter 3). Therefore, aside from the humans involved, and the activities they perform, we also observe that they are assisted, and influenced by, various non-human interactors - such as tools and processes. Thus, and given the importance of both of these in a V&V context e.g. [Wiegers, 1996, Bach, 1997, Fewster and Graham, 1999, Mosley and Posey, 2002, Burnstein, 2003, Mutafelija and Stromberg, 2003, Tian, 2005, Goodman, 2005, Nejmeh and Riddle, 2006], coupled with the influence of various organisational cultures - including information security culture [Kuusisto and Ilvonen, 2003] - we examine these additional themes within Sections 2.4 and 2.5. Further, and acknowledging the ongoing distinction between V&V theory and practice, we find that it is necessary, for completeness, to examine the associated literature exploring this distinction. This is discussed within Section 2.6 and is important given that a distinction still exists e.g. [Adrion et al., 1982, Hamlet, 1988, Osterweil, 1996, Whittaker, 2000, Bertolino, 2003, Kreeger and Harindranath, 2012, Garousi and Felderer, 2017], with this compounding actual V&V practice within industry. Finally, within Section 2.7, we present the conclusions resulting from our literature review and background analysis.

We begin by reviewing the test-based literature.

2.2 Test-Based Activities

As software testing is one of the most important, and common, approaches to V&V [Vieira et al., 2006, Ramler and Wolfmaier, 2006], we begin by examining several existing studies which focus on software testing practices within industry. Specifically, we begin by focusing on unit testing (which is the testing of the smallest possible software component [Burnstein, 2003]). As an example, [Runeson, 2006] employed a survey, following a focus group, to gauge an organisation's comprehension, strengths, and weaknesses, in terms of unit testing. Nineteen Swedish software organisations, of varying sizes (ranging from a single developer to somewhere between 100-999) were studied. As expected, unit testing was the concern of developers and not test engi-

neers (with the latter not having any say in the activity). Further, it was found that regression testing (ensuring that previously working software remains so after a change [Engström and Runeson, 2010]) was not widely practiced. Notably, few of the organisations had a unit testing process, leading the developers to define scope and thus resulting in varying implemented practices (maturity models discuss the need to establish consistency within an organisation e.g. [CMMI Product Team, 2010]). Whilst respondents believed sufficient priority was afforded to unit testing, the focus group participants indicated that the competency of those performing unit testing was very important (unit testing is generally considered the first, vital step in the testing process and, if neglected, will likely lead to harder to find defects emerging later on [Torkar and Mankefors, 2003]); they also acknowledged that it is difficult to find people with the appropriate knowledge and skills. Similarly, [Ellims et al., 2004] examine unit testing, although, in this case, across three software projects within a single organisation. They conclude that unit testing "hit[s] the spot other testing does not reach" (which they base on the observation that although unit testing was deferred until the end stages on two projects, it still uncovered defects - in some instances, the most severe - even though various reviews had been conducted throughout the earlier stages). Interestingly, [Ellims et al., 2004, Runeson, 2006] fail to consider security (even though one of the software projects studied by Ellims et al. concerns vehicle security). However, others, e.g. [Arkin et al., 2005], state the importance of considering security during unit testing.

Whilst unit testing is considered essential [Burnstein, 2003, Torkar and Mankefors, 2003, Ellims et al., 2004, Tillmann et al., 2010], it is also often misunderstood [Hunt et al., 2007]. For example, [Ellims et al., 2004] found that the attitudes of engineers, who are required to perform unit testing, change over time. They cite the example of one engineer who - although initially hostile to the idea of performing unit testing - on finding a defect, acknowledged that it would have been difficult to have found other than through unit testing. Further, many developers only generate a limited set of tests, exercising the "happy path" through the code (i.e. just establishing that correct input results in expected output, thereby overlooking unexpected input) [Hunt et al., 2007]. Additionally, although unit testing is predominately performed by developers [Andersson and Runeson, 2002, Runeson, 2006, Wojcicki and Strooper, 2006, Hashmi et al., 2010], there is an identified need to improve their motivation in this regard [Runeson, 2006] (the importance of unit testing needs to be stressed to developers [Torkar and Mankefors, 2003]). As [Burnstein, 2003] highlights, good unit testing requires preparation (preparation appears to be an unattractive task for software engineers [Kollanus, 2009]), however, in many instances, it is performed in an informal fashion [Andersson

and Runeson, 2002], with the defects found rarely being reported (whilst Burnstein considers this poor practice, [Tian, 2005] indicates that this is because unit testing forms part of software implementation).

Notably, on the successful completion of unit testing, the testing process has really just begun [Myers, 2004] (as Myers observes, it is possible to pass all the unit tests, but then fail the acceptance tests). Thus there are additional levels of testing which can, and should be, performed on a software project. This includes: integration and system testing (with the former testing the integration between components and the latter focusing on the complete system [IEEE, 1990]), as well as acceptance testing (which establishes whether the system satisfies the customer's acceptance criteria [IEEE, 2008]). Therefore, we now look at some of the existing literature concerning these levels of testing. For example, in one of the initial surveys of the New Zealand software industry, [Groves et al., 2000] aimed to understand the methods used to express requirements and specifications, as well as the V&V techniques utilised to ensure that the software being developed satisfies its requirements. Telephone interviews were conducted with 24 organisations, with the notion of acquiring a broad understanding, and face-to-face interviews were conducted within four additional organisations. As with other studies (e.g. [Runeson, 2006]), organisational size was based on the number of developers (in this instance, a small organisation has 1-3 employees, a medium organisation 4-9 and a large organisation 10 or more; notably, this does not help identify the size of the organisations, nor does it help in being able to contrast the results with other studies - hence showing the importance of adopting a common definition, see [Ayyagari et al., 2007]). Each organisation was classified in terms of reaching a certain level of testing "rigour", with the amount of time dedicated to system and integration testing varying from 9-35%. The in-depth interviews provided additional insights. For example, one organisation, employing ∼250 people, had its security systems division analysed (comprising ∼45 people, 26 of which were software developers, with each project group having a ratio of 1-3 test engineers to 4-5 developers). In this instance, software testing accounted for a third of the effort on the project. Interestingly, Groves et al. indicate that the most significant factor in determining the software practices employed within an organisation appears to be related to the size of the software development group e.g. with larger groups having more well-defined practices and also appearing to apply more software testing rigour (this emphasises the importance of considering SMEs and large organisations as distinct entities, especially since they are known to differ [Street and Meister, 2004]). Notably, across the four organisations studied in-depth, the presented results do not permit comparisons to be drawn (e.g. within one organisation there is no test-related discussion).

In a survey of the software development practices within Malaysian organisations, [Baharom et al., 2005] employ a structured questionnaire, containing 69 questions, to determine areas including: the extent that standards, processes, tools and methods are used; the software development activities; and training. Of the 41 respondents (none of which held test-related positions), it is difficult to gauge the size of the organisations reached (it is stated that a high-percentage come from small IT organisations). However, a variety of industry sectors were represented, with an almost equal divide between government and the private sector. Based on this divide, the use of test tools was approximately 20% and 35% respectively. Further, whilst there was an awareness of formal testing, the majority of organisations only applied formal testing towards the end of a project (the cost of addressing defects increases as a project nears completion, see Section 1.3.4). Overall, each surveyed organisation suffered at least one quality-related problem (similarly, [Andersson and Runeson, 2002] found, within all the organisations they conducted interviews, that problems and difficulties existed with their V&V processes), including: that the software released required further improvement (75%), that the organisation was unable to deliver the software on time (55%), budgetary problems (20%) and unhappy customers (22.5%). Notably, improving an employee's knowledge and skills was deemed important - with experience considered the most effective approach in terms of improving the situation, followed by training and formal education. However, the majority of the organisations failed to adopt any form of standard or governing process and simply continued with existing practice. This was attributed to a lack of awareness (see Section 2.6 for further discussion on the divide between V&V theory and practice).

Similarly, [Sung and Paynter, 2006] used an online survey to understand the software testing practices within the New Zealand software industry. Sixty-one responses were received from a variety of sized organisations (45% had between 1-9 employees, with some over 10,000). Sung and Paynter make several observations in terms of the smaller organisations surveyed, namely, that they do not fully embrace software testing methodologies due to time and cost constraints, as well as a lack of expertise. However, and regardless of organisational size, the adoption of tools and standards, and the provision of training, was found to be limited. Further, 61% of the respondents claimed little or no training (18% claimed to have received considerable training and, whilst security testing was referenced, there was, unfortunately, no elaboration on this). Significantly, within the smaller organisations, the majority of the training received was either on-the-job or self-directed. Fifty-one percent of the organisations did not follow any form of testing standard, however, 41% followed in-house standards. Sung and Paynter speculate that existing standards might not fit well within New Zealand or-

ganisations (which emphasises the need for adopting a specific national context) or that their existence was overlooked. Notably, they indicate that organisations may be able to cope without standards when there are a small number of employees. In summary, many of the small organisations overlooked key aspects of software testing, it was also highlighted that testing, as an activity, did not receive the recognition it deserves (in contrast, [Andersson and Runeson, 2002] found that test engineers, within both small and large organisations, received both recognition and rewards for their contribution to product quality). Sung and Paynter attribute this to the inevitable need for employees to be multi-skilled, as well as to organisations not separating out the testing role (developers were primarily responsible for testing within the smaller organisations, with 39% of the total surveyed organisations not having a separate testing role).

More generally, and given the importance of regression testing, [Engström and Runeson, 2010] conduct one of the first surveys to understand regression test practices within industry. Specifically, they held an initial focus group (involving 15 participants) and then employed an online survey (which received 32 responses to the 45 questions posed) to validate the findings. Organisational size was determined by the number of developers, which ranged from 1-999 in number. They aimed to establish: regression test definitions and practices, the challenges faced and improvement suggestions. Interestingly, although there was agreement found in terms of what constitutes regression testing (security was not considered), the aims of such testing were found to not only differ between organisations, but also within organisations (as before, a lack of consistency in practice is considered a sign of a less mature practice [CMMI Product Team, 2010]). Notably, they outline that industry practice is mostly based on experience and "not on systematic approaches". Whilst regression testing was performed at the different test levels (e.g. at the component and system levels), and at different points within the adopted software development lifecycle, the majority of the participants reported that a lack of time and resource, as well as insufficient tool support, impacted an organisation's regression testing. However, as Engström and Runeson indicate, many of these challenges are not specific to regression testing, but are applicable to software testing in general (this is echoed by others e.g. [Park et al., 2008, Nagy and Víg, 2008, Larusdottir et al., 2010]).

Whilst [Andersson and Runeson, 2002] indicate that current V&V practice, within industry, is seldom studied, we find this is particular so in terms of security V&V. However, we now examine several studies where security testing has been touched upon. For example, [Garousi and Varma, 2010] replicated the study by [Geras et al., 2004], both of which examined software testing practices in Alberta, Canada. Whilst the studies focus on software testing in general (e.g. identifying the test tools and frameworks

used), they were able to show the level of change over a five year period. Notably, the results show a significant decrease in focus on the security properties of the product under test (decreasing from over 40% to under 20%). Unfortunately, the reason for this change is not discussed. Similarly, [Causevic et al., 2009a, Causevic et al., 2009b], who conduct a large web-based survey (containing 260 questions), aimed to establish the benefits, and state, of software testing practice. Significantly, they were able to establish the degree to which security testing is practiced at the component and system test levels. Unfortunately, the reasons for the levels of practice observed were not discussed (even when focusing on the contemporary aspects of software testing [Causevic et al., 2010] - we, and others, e.g. [Rooksby et al., 2009], would consider security testing a contemporary aspect). Further, whilst [Larusdottir et al., 2010] focused on usability testing, they examine eight test techniques in total, including security testing. Utilising an online survey and interviews (encompassing 25 respondents and six interviewees respectively), the results show that the majority of organisations surveyed perform little or no security testing, however, when it is performed, we find that the majority are either developers (44%) or test engineers (33%), with the remainder including external test engineers, customers and others. Notably, a lack of training or knowledge was identified as the primary reason for why security testing was not performed as regularly as the other forms of testing analysed (in addition, security testing was the most impacted testing technique in terms of this particular reason). In a more focused study, [Epstein, 2009] conducted eight interviews, within a variety of organisations, to understand several software assurance practices. They conclude that most organisations are aware of the risks of producing insecure software and are generally motivated to produce secure software through fear of bad publicity. It was highlighted that the techniques employed vary by vendor, however, all interviewees agreed the importance of developer training in this regard. Notably, Epstein states that most of the organisations studied had centralised security functions, containing employees with relevant expertise, that could assist the development teams. However, given that Epstein defines a medium-sized organisation as having a sales volume between $100 million and $1 billion, we recall that our definition of an SME indicates that annual turnover should not exceed €50 million (see Section 1.2.1). We feel these definitions are suggestive of two very different operating environments. Similarly, whilst [Cruzes et al., 2017] examine the role of security testing within four Agile teams (based in Austria and Norway), using semi-structured interviews, organisational size is not captured. However, they aim to identify when security testing is performed, the level of associated automation, as well as the benefits of performing security testing.

Notably, it is often considered a harsh reality that software components, which op-

erate both as desired and as expected when examined in isolation (i.e. through unit testing), fail to do so when interacting with other components [Engel, 2010]. A fundamental cause for this is when one developed component makes unfounded assumptions about another component. Intercomponent and interoperability issues can be overlooked during unit testing [Gao et al., 2003]. Significantly, a frequent cause of vulnerabilities is the failure to appropriately check input values (many developers fail to check for unexpected input [Hunt et al., 2007], as do test engineers [Myers, 1978]) - this typically results through problems in integration [Allen et al., 2008]. Unfortunately, integration testing is deemed as being one of the least understood areas of software testing [Trew, 2007]. Similarly, system testing is considered as being the most misunderstood and the most difficult aspect of the testing process [Myers, 2004]. However, the literature indicates that system testing is the primary level of software testing being performed within organisations, generally followed by integration testing [Baharom et al., 2005, Larusdottir et al., 2010, Hashmi et al., 2010]. In summary, the literature above shows the importance of including both developers and test engineers within a study (especially given their differing skill sets, mindsets, personalities and mental processes [Pettichord, 2000, Sawyer, 2001, Cohen et al., 2004, Dhaliwal et al., 2011, Zhang et al., 2014, Zhang et al., 2018]), as well as ensuring a focus across the different test levels i.e. security should not be considered solely a system test activity like it is in many key V&V texts (e.g. [Burnstein, 2003, Myers, 2004]). Clearly, the cost of locating and addressing defects, and vulnerabilities, increases as a project progresses (see Section 1.3.4). However, it is from examining the different levels of testing, that we can deduce that each offers a distinct contribution (and challenge) to a software project and organisation. It is also important to keep in mind that vulnerabilities can be detected through each level of testing, as well as through different types of testing e.g. performance testing and installation testing [Stamp, 2011]. Whilst the existing literature covers various aspects of software testing, the topic of security is typically overlooked (even when the organisations being studied have a security focus e.g. [Groves et al., 2000, Ellims et al., 2004]). Thus it remains unclear, within a UK-based software SME context, how security test-based activities (i.e. security testing and penetration testing) are performed, how regularly, by whom, and how much is expended on the activities, as well as whether tools and processes covering these activities exist and are used. Further, and emphasising the need to adopt a socio-technical approach when studying V&V, [Rooksby et al., 2009] conclude, following an ethnographic study, that the problems concerning testing are not just technical, but are also organisational in nature (for example, they found that developers consider testing boring and the least enjoyable aspect of their work, see Chapter 3).

We now examine the complementary review-based activities.

2.3 Review-Based Activities

In an attempt to understand software reviews, as practiced within industry, [Kollanus and Koskinen, 2006] perform six case studies, within five Finnish software organisations (however, organisational size is not defined). Eighteen interviews were conducted and the Inspection Capability Maturity Model (ICMM) [Kollanus, 2011] was employed as the framework to analyse existing practice. In summary, all of the organisations surveyed contained weaknesses in how they practiced software reviews. This included how the reviews were implemented and focused within an organisation. For example, whilst requirements were regularly reviewed, it was quite rare to review code (which contrasts with the view of [Tian, 2005]; however, [Zheng et al., 2017] report that only about half of their 25 respondents performed code reviews, with the majority of the rest considering them as not being relevant). Further, although all but one organisation had a governing process in place, many of the interviewees knew little about these processes (similarly, when observing a series of code review sessions within an organisation with an established code review practice, [Mäntylä and Lassenius, 2009] found that although a review checklist existed, it was not used when performing the reviews, with not all employees even being aware of its existence). Kollanus and Koskinen also identified the absence of any review-based training, as well as limited knowledge of review-based activities more generally. Significantly, and regardless of organisation maturity levels, the motivation to review another's work was considered the biggest challenge (notably, within one organisation, [Baker, Jr., 1997] found that even though performing code reviews was part of an engineer's objectives and annual performance review (which runs contrary to the advice of [Fagan, 1976]), all six engineers within the survey failed to perform a code review - unfortunately, Baker, Jr. does not elaborate on why this happened; we also recall that [Runeson, 2006] found motivation problems in terms of executing unit tests, showing that V&V activities, in general, can suffer from a lack of motivation). Although Kollanus and Koskinen acknowledge that training could help increase the level of software review knowledge within an organisation, and help address issues of motivation, they do not identify the reasons for the lack of motivation. This emphasises the need to consider V&V as a set of socio-technical activities (see Chapter 3). However, [Tervonen and Harjumaa, 2004], who conducted 12 interviews within Finnish organisations with SME-sized software development departments, aimed to establish both the factors which motivate, and hinder, software reviews within industry. Specifically, they found the most important factors motivating people into performing

software reviews were: finding defects earlier, in greater quantity and quicker. The sharing of information and expertise was also viewed as important (this aspect has been echoed by others e.g. [Coram and Bohner, 2005, Mäntylä and Lassenius, 2009]). Notably, whilst many of the organisations felt process improvement was important, many emphasised that a lack of time was the most pressing impediment impacting software reviews, followed by a lack of resources. Software reviews being viewed as labourious was also an identified issue (which is also seen in terms of security review-based activities e.g. [McGraw, 2008, Edmundson et al., 2013]), as was a lack of expertise and the view that software reviews are not seen as being required (V&V activities being viewed as dispensable has been identified by others e.g. [Torkar and Mankefors, 2003] report that 60% of their survey respondents indicate that V&V was the first area to be reduced to cope with project time constraints; this is also the case within the context of small organisations [Rodrigues et al., 2010]). In conclusion, although the organisations acknowledged that software reviews could find defects earlier, in greater quality, and quicker, a shortage of time was the main obstacle preventing their adoption within SMEs (this is one of the most common perceptions surrounding software reviews [Shepard and Kelly, 2001]).

Observing that much of the research concerning software reviews involves experiments, [Kelly and Shepard, 2002] aimed to understand the qualitative issues which surround such practice. Establishing a set of maxims, they find that each review technique adopted not only has an influence on a reviewer's behaviours, but that it is also related to their past experiences, current practices, as well as their personal preferences. Similarly, [Devito Da Cunha and Greathead, 2007] attempt to understand whether a correlation exists between personality type and an individual's ability to review code. Using the Myers-Briggs Type Indicator (MBTI) assessment, they profiled 64 students and then asked them to review 282 lines of Java code containing 16 seeded defects. The only significant correlation found concerned the Sensing-Intuition factor where, those determined more intuitive, performed significantly better than the identified sensing types. Thus software organisations should capitalise on the strengths of their employees when assigning tasks, with their personality types and abilities impacting both productivity and product quality ([Myers, 1978] also found this in a software testing context). A comparable study (in that it involved 64 students reviewing Java code which had been seeded with defects) was performed by [Dunsmore et al., 2001]. In this instance, the focus of the study was to contrast two approaches when reviewing object-oriented code. Notably, the delocalised nature of object-oriented code negatively impacted the software reviews. However, as the study participants were students - with no experience in code review - this threatens the validity of their experiment (we find

this in other software review studies e.g. [Kelly and Shepard, 2002, Uwano et al., 2008], with Kelly and Shepard preventing collaborative discussion between their subjects - which, as they acknowledge, does not reflect industry practice - developers spend more time discussing code-related information face-to-face [LaToza et al., 2006]).

Also acknowledging the prevalence of using students as subjects, [Berling and Runeson, 2003] examine two software review methods within a large Swedish organisation. They employed a combination of surveys and interviews to evaluate and compare the two methods. Interestingly, the method which the respondents had the greatest confidence in, and which was found to be the most effective and efficient, was the one introduced to the organisation by those actually performing the reviews (with the other considered an "established company standard"). Berling and Runeson acknowledge that this may have influenced the level of motivation observed in terms of this particular method (anecdotal evidence suggests that experienced reviewers display considerable resistance in terms of adapting their techniques when receiving external suggestions [Kelly and Shepard, 2002]; it is also known that "people unconsciously prefer to apply existing skills with which they are familiar" [Basili et al., 1996, p. 156]). Similarly, [Uwano et al., 2008] conduct an experiment to determine the difference in performance (i.e. effectiveness and efficiency) when a subject performs checklist-based reviews on both code and design artefacts. Their aim was to establish whether good code reviewers were also good at conducting design reviews - thereby enabling tasks to be assigned appropriately or training needs identified. Ten subjects (nine graduate students and one academic, two of which had industry experience) were asked to first locate nine defects seeded within four documents and then eight defects seeded within some C source code. Whilst a correlation was expected in terms of a subject's defect find rates (i.e. a reviewer finding more defects in the design review, will also find more defects in the code review), no correlation was found i.e. good code reviewers are not necessarily good at design reviews.

However, to determine which properties are deemed the most important during code review, [Nelson and Schumann, 2004] surveyed NASA, and the aerospace industry, to establish whether a subset of software properties exist that, if met, lead to the code being deemed trustworthy. Given that a large number of properties exist - not all of which can be realistically checked - it is important to establish which should be prioritised. Notably, Nelson and Schumann highlight that such a decision is often arbitrary and dependent on the expertise and experience of the reviewer. Thus they question: is code review trustworthy, or just if you trust the expertise of the reviewer? To identify the properties, Nelson and Schumann generated a diagram with two metrics: difficultly (i.e. the effort required to check a property) and importance (i.e. frequency

and risk). Unsurprisingly, following the survey, 'security properties' appears within the high-difficultly and high-importance quadrant (finding vulnerabilities is complex [Austin and Williams, 2011]). The results also emphasised that whilst code reviews detected approximately 50% of the located defects, additional V&V tools and techniques are required to ensure trustworthy code. Additionally, we, like [Mäntylä and Lassenius, 2009], find that much of the research concerning code reviews focuses on defect counts as opposed to defect types (similarly, [Bertolino, 2003] indicates that several software testing studies focus on the number of detects as opposed to their impact). This is an important distinction since, as Mäntylä and Lassenius indicate, a focus on defect count can result in a misleading view e.g. there are different types of defect with differing impacts (this was discussed, in terms of defects and vulnerabilities, within Sections 1.3.3 and 1.3.4). Similar to Nelson and Schumann, Mäntylä and Lassenius conclude that different V&V activities will find different types of defect. This prompts an examination of the relationship between test and review-based activities.

Whilst the existing literature predominately treats test and review-based activities in isolation (as evidenced above), several studies consider both (as [Aurum et al., 2002, Ciolkowski et al., 2002] question: what is the relationship between these activities, what types are defect are best found by each and how do they complement one another). These studies typically take the form of controlled experiments e.g. [Myers, 1978] studies the relative effectiveness, as well as the influencing factors and associated costs, of three different approaches when testing and reviewing a small text-formatting program which was seeded with 15 errors. Notably, and given that the subjects were highly experienced programmers, only a third of the seeded defects were found. Myers concludes that both test and review-based activities should be employed together on a project. Similarly, [Wood et al., 1997] conduct an empirical study comparing the effectiveness and efficiency of three approaches to defect detection (this replicates [Myers, 1978], as well as at least three other studies (e.g. [Hetzel, 1976, Basili and Selby, 1987, Kamsties and Lott, 1995]), however, only [Myers, 1978] utilises practitioners, with the others relying on student subjects). Given the absence of any consistent evidence, in terms of showing which approach is best, Wood et al. conclude that the approaches appear to be complementary. Interestingly, [Porter et al., 1998] found, when examining the application of unit testing before code review (within a large organisation), that whilst it would have been expected that the amount of defects found by the code review would have decreased as the amount of pre-review testing increased, this was not observed ([Mäntylä and Lassenius, 2009] observe that quality assurance applied before code review can have a significant impact on the results, with [Kelly and Shepard, 2002] indicating that "[u]nit testing before inspection is contrary to some accepted wisdom").

Notably, whilst [Myers, 1978, Wood et al., 1997] do not consider vulnerabilities, [Austin and Williams, 2011] state that no single tool or technique is sufficient in locating all vulnerabilities or even all types of vulnerability. Collectively, these studies emphasise the importance of organisations embracing more than one approach to defect detection (a view echoed by others e.g. [Adrion et al., 1982, Bertolino, 2003]).

The importance of software reviews has long been acknowledged [Porter et al., 1998, Aurum et al., 2002, Dunsmore et al., 2001, Kollanus and Koskinen, 2006]. Clearly, an inherent strength is in their being able to be applied to any type of artefact (e.g. improving the quality of software documentation reduces development time [Uwano et al., 2008]). They can also be employed to uncover defects and vulnerabilities throughout software development (clearly leading to a cost benefit when discovered earlier, see Section 1.3.4). A successful review depends on several factors, including: possessing an understanding of the product and the technologies involved, the use of appropriate tools, as well as previous experience [Parnas and Lawford, 2003]. Notably, it is important that those performing the review are familiar with the product, the application domain and the process of conducting a review itself [Aurum et al., 2002, Kelly and Shepard, 2002] (otherwise, according to Aurum et al., they must be trained; [McDermott, 2000] discusses the importance of product domain knowledge when performing security V&V activities). Notably, reviewers vary in their ability to detect defects, which is possibly due to individuals looking for different types of issue [Porter et al., 1998] (the importance of conducting a focused security review has been emphasised [Meier, 2006]). As stated in [Mäntylä and Lassenius, 2009] (and supported by the literature reviewed), there is very little information on the types of defects detected by software reviews e.g. there is a tendency to focus on defect count. Mäntylä and Lassenius acknowledge that practitioners will be aided by understanding which V&V technique is best suited to detect which type of defect. In summary, the approach adopted, clearly relies on the awareness and the ability of the reviewer (we examine the importance of education, and the distinction between theory and practice, in Section 2.6).

However, we observe that software reviews are not always applied in practice [Aurum et al., 2002, Kollanus and Koskinen, 2006], with [Ciolkowski et al., 2002] indicating that many organisations have not taken full advantage of them. This is attributed to people issues causing too many complications (which includes expertise and communication problems) [Kelly and Shepard, 2002]. Significantly, [Kollanus and Koskinen, 2006] found that software reviews do not appear to be a pleasant task for software engineers [Kollanus and Koskinen, 2006] (they have been considered labourious and time-consuming [Tervonen and Harjumaa, 2004, Edmundson et al., 2013]). This em-

phasises the importance of adopting a socio-technical approach when studying V&V (see Chapter 3). It is also apparent that organisations need to ensure that they have processes which support software reviews [Aurum et al., 2002, Kelly and Shepard, 2002, Ciolkowski et al., 2002]. Notably, the existing literature typically overlooks security reviews. Specifically, it remains unclear, within UK-based software SMEs, how security review-based activities (i.e. security design and code reviews) are performed, how regularly, by whom, and how much is expended on the activities, as well as whether tools and processes covering these activities exist and are used. Whilst review-based activities have been considered as encompassing a set of effective manual techniques [Adrion et al., 1982], they can be assisted by automated tools, thereby helping increase the efficiency of a review and saving both time and effort [Aurum et al., 2002, Parnas and Lawford, 2003, Nelson and Schumann, 2004, McGraw, 2008, Mäntylä and Lassenius, 2009]. We thus now examine the use of tools and frameworks within the wider context of V&V as a whole.

2.4 Tools and Frameworks

Knowledge of tools and frameworks plays an important role in developing high-quality software products [Tian, 2005]. Thus, it is important to examine these, especially since many V&V tools and frameworks exist [Zhu et al., 2006, Budnik et al., 2007, Clarke et al., 2010]. By employing these, organisations are able to automate a number of otherwise manual activities. Much has been written about test automation [Bach, 1997, Fewster and Graham, 1999] - with some extolling the benefits (such as repeatability, consistency and efficiency), some the pitfalls (such as unrealistic expectations, maintenance requirements and lack of organisational support) - however, it is commonly accepted that manual and automated tests find different classes of defects and, therefore, should be deemed complementary [Mosley and Posey, 2002]. We thus examine the literature concerning tool and framework adoption.

Wicks surveyed several software organisations, based in Scotland, to understand the tools used during software development [Wicks, 2005]. Of the 33 respondents, from various industry sectors, approximately 72% worked within organisations employing less than 100 people. Thirty questions were posed, ranging from understanding aspects of the organisation, to the programming languages and tools utilised. Notably, there was only a single instance of a test tool being integrated within an organisation's software development process; however, there were indications that a small minority were making use of some test-related tools (significantly, ∼64% of the respondents had not received training on the tools adopted). The continued lack of test tool adoption

is interesting, since a similar lack was reported in [Zelkowitz et al., 1984] e.g. of the 30 organisations surveyed, only 27% used some form of test tool. Neither study considered any form of security-related tool.

In a more focused survey, [Nagy and Víg, 2008] elected to understand the development environment, test approach and tools utilised on Erlang (a general-purpose programming language) projects. Comprising 21 questions, 200 responses were received. Significantly, of the 14 Erlang tools discussed, and in all cases, the respondents were more aware of their existence, over their actually being used. These tools included test frameworks, static analysis tools and stress testing tools. Notably, no freely available, stable test tool existed. This has implications, since SMEs are keen to exploit free software to assist with software development [Andersson and Runeson, 2002, Sitnikova et al., 2007]. As Nagy and Víg conclude, testing is poorly supported on Erlang projects, with most developers using proprietary tools or manual approaches to both unit and system testing. Continuing with SMEs, [Thörn and Gustafsson, 2008] found tool support as being both cumbersome and expensive within their survey of 27 Swedish SMEs (atypically, this survey adopts the same SME definition as us, see Section 1.2.1). Interestingly, they outline that software modelling, as a discipline, is covered in software engineering courses (this can be contrasted to V&V more generally, see Section 2.6), that there is mature tool support, and that a wealth of domain knowledge and practical experience exists to be drawn upon, therefore, it might be safe to assume that model-based software engineering is common in practice (such an assumption appears relatively sound). Unfortunately, a conclusion drawn is that the constraints faced by SMEs preclude the application of software modelling, generating the view that it as a "considerable overhead". Given that security V&V is an empirical unknown (see Section 1.3.6 and [Kreeger and Harindranath, 2012]), we cannot make such an assumption.

Berner et al. present an experience report following their observation of 12 software projects over a three-year period [Berner et al., 2005]. This encompassed understanding an organisation's approach to test automation and the creation of testware. Notably, whilst it was found that certain tests are not effective, or possible, without automation, they observed an improvement in both the quality and depth of the test cases when test engineers were "freed from boring and repetitive tasks". Specifically, this permitted time to be spent on designing new and better tests, as well the ability to focus on neglected areas and difficult tasks (we recall the complexity of security testing [Thompson, 2003, Austin and Williams, 2011]). Unfortunately, no specific examples were given, however, time constraints are known to impact most types of testing - with security testing being no exception [Larusdottir et al., 2010] - and when test time is

reduced, scope is as well [Gilbert, 2011]. Similarly, [Kasurinen et al., 2010] also examine test automation practices (employing both a survey and interviews, across 31 and 12 organisations respectively). Specifically, they focused on how organisations have adopted test automation, the issues faced and how software testing has been impacted. Interestingly, they found that test automation is usually introduced due to "pressure to create resource savings", however, the amount of test automation found was considered less than expected based on the existing literature. These findings apply to both SMEs and large organisations. Notably, 73.8% of the organisations surveyed by [Sung and Paynter, 2006] employed manual testing, with only 39.3% performing any form of automated testing (significantly, whilst the majority of the organisations studied by [Andersson and Runeson, 2002] expressed a desire to increase the amount of test automation being performed, a lack of time often prevented this). Sung and Paynter also observed that 56% of the organisations did not utilise automated test tools, however, of those which did, approximately half used tools developed in-house (Sung and Paynter speculate that a high learning curve and tool cost may be the respective reasons for these values). In contrast, [Itkonen et al., 2009] focus on manual testing practices, within four Finnish software organisations, with the view of improving the understanding of how experienced-based and exploratory manual system testing is conducted. This was a qualitative study based on field observations and, although the organisations were classified as being SMEs, no definition is provided. This study shows that test engineers employ various techniques and strategies during test execution and do not necessarily rely on test documentation. It was also highlighted that most new defects are found by manual testing, with test automation removing the simple, repetitive tasks, thus permitting time for creative manual testing (which we, like others, e.g. [Türpe, 2008, Michael et al., 2013], view as encompassing security testing). As indicated by Itkonen et al., test execution is not simply a mechanical task, but involves skill and knowledge (it has been shown that this has as "strong an effect on the results of testing as do the test case design techniques" selected [Itkonen et al., 2009, p. 494]). This is further supported by Bertolino, who states that: "[a]lthough there is much room for automation [...] the tester's expertise remains essential" [Bertolino, 2003, p. 17]. This aspect is discussed further in Section 2.6.

Notably, many organisations have unrealistic expectations about test automation (and the associated benefits and challenges faced) [Berner et al., 2005]. Whilst employing such technology can minimise some of the drudgery associated with testing [Myers, 2004], there are instances where it is inappropriate, disadvantageous and expensive, as well as situations where it is appropriate, advantageous and cost effective [Everett and McLeod, Jr., 2007]. A common misunderstanding with the automation of manual tests

is that more defects will be found, however, this is considered doubtful [Fewster and Graham, 1999]. For example, [Ellims et al., 2004] found that the design of unit tests identified more defects than the execution of the tests; further [Berner et al., 2005] report that manual tests detect most of the new defects, not the automated tests. There is always an economic trade-off to be considered between manual and automated testing [Ramler and Wolfmaier, 2006]. Significantly, [Tervonen and Harjumaa, 2004], within an SME context, found difficulty in introducing tool adoption within organisations, stating: "companies strive to keep their development process alive and try to avoid all disturbing factors". Thus they consider it a challenging task to encourage SMEs to use new tools, observing that such attempts are likely to fail if an established process does not exist (even if it is a "very good tool", process integration issues can prevent tool adoption [Favre et al., 2003]). Notably, although tool adoption is a major concern in software engineering, the barriers to adoption are not just technical in nature, but also organisational e.g. [Favre et al., 2003] highlight that the size of an organisation plays a significant role in tool adoption (as it does in process adoption [Staples et al., 2007]). This emphasises the need to consider V&V as a socio-technical activity (see Chapter 3).

Aside from determining whether it is appropriate to adopt a test tool, or to automate an activity, it is important to consider the process of tool selection and evaluation itself [Poston and Sexton, 1992, Bach, 1997, Illes et al., 2005]. This is especially relevant when considering that test tools are often designed without adequate attention [Nakagawa et al., 2007] (this incompleteness has plagued tool adoption for some time [Zelkowitz et al., 1984]). Furthermore, it is as well to recall the maxim: "a fool with a tool is still a fool" and that the adoption of tools requires both training and organisational support. The survey performed by [Wicks, 2005] shows that test tool adoption, and the associated training, was low; further, [Nagy and Víg, 2008] highlight, in some instances, that awareness of a tool's existence was more prevalent than its actual deployment, with tool support and maintenance, more generally, being seen as both burdensome and expensive. The industry need for efficient and effective tools continues to grow [Vieira et al., 2006], and given that no one tool can assist all V&V activities [Adrion et al., 1982, Everett and McLeod, Jr., 2007], Adrion et al. indicate that this implies the acquisition and learning of several tools - which is something that can incur more costs than the tools themselves. In summary, whilst the existing literature shows a lack of tool support and adoption, across a variety of organisations, it remains unclear as to whether UK-based software SMEs employ V&V-related tools (and if not, why not), the origin of these tools and whether they are customised to their particular environment, as well as the extent to which automation technologies

are employed. Given the relationship between tools and processes, as well as the level of V&V maturity within an organisation, we now examine these aspects.

2.5 Organisational Maturity, Processes and Culture

Software organisations rely upon processes to develop products [Wiegers, 1996, Burnstein, 2003, Mutafelija and Stromberg, 2003, Goodman, 2005, Nejmeh and Riddle, 2006] (a process is "a structured set of activities and decisions to do a certain job" [Dybå et al., 2004, p. 2]). Further, organisations are known to examine standards (to understand "why" their processes should exhibit certain properties), maturity frameworks (to identify "what" activities should be captured by their processes) and best practices (to determine the optimal approach i.e. "how" the identified activities should be performed) [Nejmeh and Riddle, 2006]. Notably, successful organisations continuously manage and strive to improve their processes [Mutafelija and Stromberg, 2003, Nejmeh and Riddle, 2006, Harrington, 2006]. However, in terms of which process to adopt, organisations are faced with the decision of "pret-á-porte vs. tailor-made" [Ward et al., 2001, p. 106], as well as a choice from a "dizzying array of software and system process standards, recommended practices, guidelines, maturity models, and other frameworks" [Sheard, 2001, p. 96] (see [Paulk, 2004] for a list concerning software organisations). In terms of V&V, as an organisation increases its level of test maturity, the testing process itself becomes more defined, managed, and measured, in turn, leading to the potential for improved software product quality [Burnstein, 2003] (as [Hartman, 2002] found, organisations with a higher maturity level also aim to remove defects earlier within the software development lifecycle) and, as acknowledged by [Brown and Duguid, 2001]: practice without process will likely become unmanageable, however, process without practice will result in the loss of creativity which is required for sustained innovation. Given the relationship between an organisation's maturity and processes - and their success - as well as that employees are influenced by various organisational cultures [Kuusisto and Ilvonen, 2003], we now examine the literature in terms of V&V maturity, process, and information security culture.

We begin by examining [Koomen, 2002, Kollanus, 2009], both of which adopted test and review-based maturity models in which to conduct and structure their respective studies. In order to understand the use of the Test Process Improvement (TPI) model, [Koomen, 2002] conducted a global web-based survey. The TPI model provides guidelines for assessing an organisation's test maturity level and guidance to organisations on how to improve [Koomen and Pol, 1999]. Unfortunately, although Koomen contrasts small and large organisations in their analysis, they define a small

organisation as having less than 1,000 employees. However, within the majority of the organisations (some 73%), a separate test function exists (indicating an organisation with a higher test maturity level). Koomen also observes that the smaller organisations typically have less formal processes in place. Similarly, [Kollanus, 2009] conducted a series of interviews, within seven Finnish organisations (of varying sizes), to assess the maturity of an organisation's software review practices, as well as to identify weaknesses. They employed the Inspection Capability Maturity Model (ICMM) [Kollanus, 2005, Kollanus, 2011] to assess maturity (the five levels within this model correspond to those within [CMMI Product Team, 2010]). Notably, all of the organisations assessed were classified at the lowest maturity level (although some practices satisfied elements from the next two levels). This implies that none of the organisations either monitor, or collect, information on their review process (see [Kollanus and Koskinen, 2006]). A further weakness, observed across all of the organisations studied, concerned the lack of review-based training received. Further, there was variation in how the review activities were perceived e.g. whilst the review of test cases was clearly perceived as being important, attitudes varied in terms of code reviews. Unfortunately, the reasons for these views were not explored.

In a study of Swedish software organisations, ranging in size from 15-2,000 employees, [Grindal et al., 2006a, Grindal et al., 2006b] found a low level of software test maturity. Twelve interviews were conducted, based around six questions, to determine: the test case selection methods adopted; whether the software testing being performed is guided by a test strategy; when test engineers are first involved in a project and their general knowledge; the amount of project resource dedicated to software testing and the test metrics collected and used. Notably, 50% of the organisations surveyed involved test engineers at a project's outset, ∼33% during the requirements phase (we recall the benefits of earlier defect detection, see Section 1.3.4). Whilst there was variation in time spent on software testing, an overall mean of 35.1% was reported. The majority of this time is spent on system test activities, which is attributed to a lack of test maturity and a lack of test process. Interestingly, Grindal et al. speculate that a "low level of test maturity may be safer with small projects than with larger projects", however, we feel this generalisation is inappropriate, since the deployment context of the product should be taken into account. Similarly, [Park et al., 2008] conducted a survey examining the level of software test maturity within seven Korean defence organisations (organisational size is not defined) with the view of developing a new test maturity model. The survey had three objectives: firstly, to understand the approximate level of software test maturity; secondly, to understand the software testing practices used; lastly, to identify problems specific to software development in

the context of defence. Thirty-eight responses were received and test maturity was evaluated using TPI [Koomen and Pol, 1999]. The V-Model was found to be the dominant software development methodology adopted, with unit and integration testing the predominate types of testing being performed (which is probably reflective that the majority of respondents were developers). Notably, the majority of respondents believed their testing was neither sufficient or insufficient, however, there was a common theme reported in that a lack of time, and a lack of test resource, impacted the activities being performed.

Further, and continuing the defence industry theme, [Sitnikova et al., 2007] performed structured interviews within 16 South Australian defence organisations (almost 69% of which fit within our definition of an SME) to determine the processes, standards and tools utilised within a software engineering context. They also aimed to understand the strengths and challenges faced. Notably, the smaller organisations reported investment constraints in terms of process improvement (this aspect was considered an overhead, however, whilst [Ward et al., 2001] consider the cost of quality as being free in the long run, they also observe that small organisations cannot ignore the associated up-front costs). Similarly, the expense of commercial tools was seen to prohibit their uptake within these organisations. Therefore, many sought to reduce their costs by using free (or low cost) open source software to assist with product design and development (certainly, tool cost is a noted barrier to their utilisation within small organisations [Rodrigues et al., 2010]). Further, whilst the majority of the organisations identified that their employees' experiences and capabilities were critical to achieving organisational success (as echoed by others e.g. [Dybå, 2003, Harrington, 2006]), the majority of the smaller organisations indicated that their main approach to ensuring their employees possessed the appropriate skills was via careful recruitment. These organisations also relied upon on-the-job training to minimise costs.

Within seven small Brazilian organisations (organisational size is not obviously defined), [Rodrigues et al., 2010] identify 17 factors, predominately from within the literature, which make test process implementation difficult. Based on these factors, they constructed a questionnaire to evaluate the impact of each factor on the institutionalisation of a test process. Notably, knowledge of test process models, such as TPI, was limited (whilst there was a degree of awareness, no organisation actually employed them). Rodrigues et al. identified the following factors, from a management perspective, which negatively impacted test process implementation: $\sim$71% of the organisations indicated a lack of time, $\sim$57% consider the test process dispensable, and $\sim$43% indicate a lack of tool support, restricted size of the test team and a lack of management commitment. In addition, the following factors were identified by the test

analysts surveyed: ∼76% of the organisations consider the test process dispensable, ∼65% identify a lack of management commitment, as well as a lack of planning and/or appropriate methodology, and ∼47% identify a lack of skilled resource. Therefore, Rodrigues et al. conclude that few small organisations are in a position to perform software testing given the difficulties faced. Additionally, we find that such factors also influence the software review activities. For example, whilst [Porter et al., 1998] indicated that most organisations follow a three-step process of preparation, collection and repair when performing software reviews, [Tervonen and Harjumaa, 2004] found, within software SMEs, that tight project schedules can prevent process improvement in this regard. We also acknowledge that no software review process is perfect [Parnas and Lawford, 2003] and that their application can vary within organisations [Aurum et al., 2002]. That it is important to tailor a process to the local environment has been known for some time e.g. [Ginsberg and Quinn, 1995]. For example, rather than adopting an approach to software reviews as captured within the literature, [Baker, Jr., 1997] tailored the process to their environment (we discuss the divergence between the literature and actual practice within Section 2.6); [Shepard and Kelly, 2001] also emphasise the importance of tailoring software reviews to a local environment.

Notably, it is hard to imagine any organisation which cannot be improved [Mutafelija and Stromberg, 2003] (for example, within a small organisation, [Pyhäjärvi et al., 2003] observed that software testing must become more integrated with the software development process). Therefore, a variety of test maturity models and software process improvement models exist [Swinkels, 2000, Basri and O'Connor, 2010] ([Conradi and Fuggetta, 2002] observe that more than 250 software standards have been proposed, however, they question how many are used in practice). In the context of test maturity models, [Staab, 2002] indicates that the majority have never found much acceptance given the limited availability of non-theoretical documentation ([Basri and O'Connor, 2010] also observe the impact caused by a lack of detailed implementation guidance). Therefore, whilst [Basri and O'Connor, 2010] indicate that quality-focused processes, and standards, are both maturing and gaining acceptance within many organisations (this increased interest is echoed by [Conradi and Fuggetta, 2002]), they also indicate that many are not being adopted by SMEs. Notably, many small organisations have encountered problems in terms of improving their software processes [Hauck et al., 2008]. For example, within an SME context, [Tervonen and Harjumaa, 2004] indicate that a lack of knowledge can impact process improvement and [Sitnikova et al., 2007] found that the limited resources of an SME can significantly impact their ability to invest time in organisational development activities such as process improvement. Further, many small organisations consider the formality and discipline of some process

models as being unduly burdensome [Ginsberg and Quinn, 1995] (this complexity can restrict a small organisation's ability to tailor and successfully implement a process [Basri and O'Connor, 2010]), which is understandable given that some of the standard models were initially developed for larger organisations (e.g. CMM, the Capability Maturity Model), therefore, complicating their use within smaller organisations [Habra et al., 1999, Basri and O'Connor, 2010]. This is emphasised by the survey performed by [Brodman and Johnson, 1994] (which found that small organisations experienced difficulty when implementing CMM) and, similarly, [Staples et al., 2007] found that small organisations were unlikely to adopt the Capability Maturity Model Integration (CMMI) model due to their organisational size. Notably, small organisations are also viewed as being more concerned about practice, with larger organisations appearing to be more concerned about formal processes [Dybå, 2003].

Although CMMI is considered the best known, and most common, reference model, it is deemed too general to provide detailed guidance on improving specific areas [Kollanus, 2009]. We would consider security V&V as falling under such an area. However, whilst several secure software development processes (e.g. Touchpoints [McGraw, 2006] and Microsoft's SDL [Microsoft, 2020]) and software security maturity models (e.g. BSIMM [McGraw et al., 2016] and SAMM [OWASP, 2017]) exist, many of these are not considered suitable for SMEs e.g. [Futcher and von Solms, 2008, De Win et al., 2009] (notably, [Kettunen et al., 2010] found, within small organisations, that guidelines and practices were observed, and adhered to, if based on in-house standards). Clearly, there are fundamental differences in terms of small and large software organisations [Dybå, 2003]. Unfortunately, whilst the existing literature provides some indication of software V&V maturity, within both small and large organisations, encompassing test and review-based activities, there is an absence of focus in terms of security V&V: specifically, within UK-based software SMEs. Thus, within this context, it remains unclear as to the level of maturity reached by these organisations in terms of their software and security V&V practices (as well as the relationship between the maturity levels reached). Also, it is unclear whether these organisations have processes covering the V&V activities and, if so, what the origins of these processes are (e.g. are they based on existing standards). As [Sitnikova et al., 2007] observe, it is necessary to identify the process improvement approaches that are suitable for smaller organisations.

Further, as various cultures are known to influence individuals within organisations [Kuusisto and Ilvonen, 2003], we aim to understand whether an organisation's information security culture influences their security V&V practice. Such a culture encompasses "all socio-cultural measures that support technical security methods, so that information security becomes a natural aspect in the daily activity of every employee"

[Schlienger and Teufel, 2003, p. 46]. A socio-cultural approach developed since security incidents were still occurring - regardless of the technology employed by organisations - as humans were still involved [Connolly and Lang, 2012] (we recall that security is a socio-technical subject [Coles-Kemp and Hansen, 2017]). Notably, an information security culture exists "when every participant in the information society, appropriately to their role, is aware of the relevant security risks and preventative measures, assumes responsibility and takes steps to improve the security of their information systems and networks" [BIAC and ICC, 2004, p. 3]. Whilst such a culture is relatively recent [Ruighaver et al., 2007], as is its investigation [Williams, 2009], many organisations have established information security awareness programmes [Kruger and Kearney, 2006]. However, although organisations now consider the development of an information security culture important [Connolly and Lang, 2012], employees are viewed as being unconsciously incompetent in terms of information security practices [Thomson and von Solms, 2006], with organisations still requiring guidance on establishing an information security-aware culture [Da Veiga and Eloff, 2010].

Given this importance, various means exist for understanding the state of an organisation's information security culture e.g. [Schlienger and Teufel, 2003, Dimopoulos et al., 2004, Kruger and Kearney, 2006, Burns et al., 2006, Dojkovski et al., 2007, Ngo et al., 2009, NIST, 2014, Da Veiga and Eloff, 2010, Department for Business, Innovation & Skills, 2013]. However, there is no one accepted approach to its study [Schlienger and Teufel, 2003, Okere et al., 2012]. Further, whilst [Connolly and Lang, 2012] define a weak information security culture as one where employees exhibit non-compliance towards the security practices stipulated by their organisation (which can result in financial losses or loss of reputation), and a strong information security culture when employees follow their organisation's prescribed security practices, much of the information security culture literature fails to provide a clear definition of what is meant by security culture [Ruighaver et al., 2007]. Additionally, many information security culture studies overlook the unique characteristics of SMEs, as well as national context [Dojkovski et al., 2007, Dojkovski et al., 2010] (they also tend to cover a broad range of industry sectors e.g. [Burns et al., 2006]). Therefore, although several SME-focused information security culture studies exist, our specific organisational and technological contexts remain unclear, in particular, we do not see how information security culture, within UK-based software SMEs, influences an organisation's security V&V practice.

2.6 Distinction Between V&V Theory and Practice

The distinction between theory and practice, in terms of software engineering and V&V, has long been acknowledged [Adrion et al., 1982, Zelkowitz et al., 1984, Hamlet, 1988, Osterweil, 1996]. Unfortunately, this gap, between industry practice, and academic research, is still in existence [Whittaker, 2000, Bertolino, 2003, Garousi and Felderer, 2017]. For example, there are several areas which have received significant attention by researchers, but which have had limited industrial application e.g. fault-based testing and test case selection [Bertolino, 2003, Briand, 2010]. Bertolino also details problems evident to practitioners, but which have largely been ignored by researchers. Similarly, although extensive research into code reading strategies exists, there has been little investigation in terms of applying such strategies to object-oriented code [Dunsmore et al., 2001] (this is the predominate programming paradigm in modern software development [Greanier, 2004]). Further, whilst there has been significant research into software reviews more generally, there are still many unresolved issues [Aurum et al., 2002]. Conversely, when practitioners lack knowledge of existing research, "they are in no position to absorb new findings from researchers" [Lethbridge et al., 2007, p. 14]. Therefore, there is clearly a gap between V&V research and practice.

Notably, it is the transfer of research to industry which is considered as being at the heart of this gap [Ivarsson and Gorschek, 2011] (whilst such technology transfer is difficult in general, it has been deemed particularly challenging in terms of software quality, with the "yawning chasm", between research and practice, being both "wide and increasingly inculturated, to the detriment of both communities" [Osterweil, 1996, p. 746]). Thus, although Ivarsson and Gorschek consider the use and adoption of research within industry as being one of the main objectives of software engineering research, it is also apparent that the research community is ahead of industry practice [Hartman, 2002]. Specifically, Hartman references the need for researchers to address "real industrial problems". This is important, since very little of the knowledge developed by the research community is seen to impact practitioners (the knowledge contained within papers at the premier software engineering conference, ICSE, is cited as an example; there is also a decline in the number of industrialists attending such conferences, thereby reflecting the different interests which exist between the two communities) [Lethbridge et al., 2007]. Therefore, whilst there is some reported use of the research literature within industry (e.g. [Wojcicki and Strooper, 2006]), practitioners are known to routinely disregard it under the belief that it is irrelevant to them [Parnas and Lawford, 2003] (Parnas and Lawford also highlight that researchers tend to write for one another, therefore, losing contact with the realities practitioners face).

Unfortunately, several perceptions, and misconceptions, continue to surround aspects of V&V (e.g. software testing has been viewed as a dispensable, boring, second-class activity that is not technically challenging and which can be addressed at the end of a project if there is time [Murugesan, 1994, Bertolino, 2003, Desikan and Ramesh, 2006, Rooksby et al., 2009, Rodrigues et al., 2010]). Further compounding this situation, we find that not only have some of these perceptions and misconceptions been conveyed by academics [Desikan and Ramesh, 2006], universities also fail to provide a sufficient focus on V&V - even though educators should create a curriculum that incorporates research to improve software engineering practice [Ardis and Mead, 2011]. For example, software testing is clearly an underemphasised element in the undergraduate curriculum e.g. [Murrill, 1998, Jones, 2000, Jones and Chatmon, 2001, Leska, 2004, Scott et al., 2004, Kazemian and Howles, 2005, Elbaum et al., 2007, Kreeger, 2009, Bajaj and Balram, 2009]. As [Jones, 2000] highlights: although software testing accounts for 50% of a project's cost (see Section 1.3.2), it receives little attention in most curricula (similarly, [Kreeger, 2009] found a lack of dedicated software testing courses within the UK education system). Further, whilst [Jones and Chatmon, 2001] consider it impractical to teach everything about software testing, they observe that it is possible to teach the necessary attitudes and skill sets to address the issues surrounding software quality (thereby emphasising the need to consider V&V as a socio-technical activity, see Chapter 3). Ultimately, the impact of failing to adequately prepare those entering the workforce is highlighted in the survey performed by [Scott et al., 2004]. Specifically, the majority of graduates, within the surveyed South African information technology-focused organisations (ranging in size from 30 to more than 1,000 employees), were unable to adequately perform software testing. It is thus necessary to strengthen the relationships between industry and academia in order to communicate real-world practices to students [Thompson and Edwards, 2008].

Given the costs associated with performing V&V (Section 1.3.2), and when failing to perform the activities adequately (Section 1.3.4), the findings above are surprising. An employee's knowledge clearly impacts their ability to adopt certain V&V techniques. For example, less formal techniques are often considered more accessible to practitioners [Adrion et al., 1982, Mařík et al., 2000, Vieira et al., 2006]. Certainly, the difficultly of formalising even a simple program often leads to "less elegant" software testing being practiced [Hamlet, 2010]. As Parnas and Lawford note: in "theory", we can prove that programs are correct, however, in "reality", this is "rarely practical and even more rarely done" (they also state that most research papers on verification make simplifying assumptions which are not valid for real programs) [Parnas and Lawford, 2003, p. 16]. In summary, there is a clear relationship between the success, in terms

of adoption and application of specific V&V techniques, and an employee's knowledge, competency and experience [Baharom et al., 2005, Runeson, 2006, Sung and Paynter, 2006]. This relationship also has a clear impact on the quality of the products being developed by organisations [Lethbridge et al., 2007].

Although connecting theory with practice should be considered the foci of any form of engineering [Parnas and Lawford, 2003], a common limitation concerns the detachment between software engineering research and practice [Egorova et al., 2009]. Ultimately, such research should support practical software development [Sjøberg and Grimstad, 2010]. However, we find that software testing, as an activity, is approached with the attitude of something that can be performed if time permits [Desikan and Ramesh, 2006]. Organisations are known to target V&V activities, in the first instance, when needing to reduce, or remove, scheduled work to cope with project time constraints [Torkar and Mankefors, 2003, Rodrigues et al., 2010]. Unfortunately, this attitude is also reflected within academia e.g. "[t]esting is one of the topics that can easily be pushed aside" [Marrero and Settle, 2005, p. 4]. Notably, practitioner knowledge of existing V&V research is considered insufficient [Osterweil, 1996, Juristo et al., 2006]. Therefore, that there is both a distinction, and a divide, between theory and practice, in terms of V&V, appears evident. Given that "[i]nnovation is one of the most important prerequisites for business success" [Rombach and Achatz, 2007, p. 30], and that discoverability problems exist with regards to grey literature and the SME community (specifically highlighted within the UK) [Swan, 2008], it is necessary to understand the training UK-based software SMEs afford to software and security V&V.

2.7 Conclusions

Having reviewed the relevant literature, it is apparent that many software engineering studies offer limited treatment of V&V. However, where V&V is touched upon, we find that these studies typically overlook security V&V (with it not being clear as to why e.g. [Groves et al., 2000] perform an in-depth survey within four organisations, one of which developed software security products, however, there was no discussion regarding security testing). Or, they focus on larger organisations: software engineering, within small organisations, is not only overlooked by the literature, but also by software engineering societies and computing institutes [Fayad et al., 2000] (notably, it has long been acknowledged that small and large organisations differ [Street and Meister, 2004]). Based on this, we can conclude that there is limited treatment, within the existing literature, as to how security V&V is performed within UK-based software SMEs (this is further emphasised by the summary provided within Appendix C). The

influence of an organisation's information security culture, on security V&V practice, also remains unclear. Further, it is apparent that much of the existing V&V literature focuses upon the technical [Rooksby et al., 2009], thereby overlooking aspects such as interaction, collaboration, co-operation and motivation. We thus now justify the adoption of a socio-technical approach when framing our study.

Chapter 3

V&V as a Socio-Technical Interaction Network

Within this chapter we show that software engineering, and V&V, are socio-technical activities. We then justify the adopted socio-technical framework. Specifically, this chapter provides:

Section 3.1: An introduction, detailing the adoption of a socio-technical perspective.

Section 3.2: Software engineering, and V&V, are examined from a socio-technical perspective.

Section 3.3: We examine several socio-technical approaches for studying technology, showing the applicability of the Socio-Technical Interaction Network (STIN) framework for studying V&V.

Section 3.4: The adoption of STIN is justified.

Section 3.5: We summarise the suitability of applying STIN when studying security V&V within software SMEs.

We begin by justifying the adoption of a socio-technical perspective.

3.1 Adopting a Socio-Technical Perspective

Prior to several field studies (e.g. [Trist and Bamforth, 1951]), technology was considered an independent, autonomous variable [Ropohl, 1999]. However, as Ropohl indicates: "[t]he concept of the socio-technical system was established to stress the reciprocal interrelationship between humans and machines and to foster the program of shaping both the technical and the social conditions of work, in such a way that efficiency and humanity would not contradict each other any longer" [Ropohl, 1999, p. 59]. It thus became acknowledged that technical and social factors interact to influence organisational outcomes [Griffith and Dougherty, 2002]. Notably, organisations are themselves considered socio-technical systems ([Cherns, 1976] considers this a tautology) which, when divided into the two sub-systems - the social and the technical - comprise: people (the social system) who utilise tools, techniques and knowledge (the technical system) to develop products valued by customers (the organisation's external environment) [Griffith and Dougherty, 2002]. We have already observed how tools (Section 2.4), the adopted techniques (Sections 2.2 and 2.3), and the knowledge of engineers (Section 2.6), impacts V&V within an organisation and, therefore, influences the shape of the final product.

Figure 3.1 shows a typical socio-technical system, with [Nelson and Quick, 2013] observing that:

- Structure: denotes the systems of communication, authority and workflow.
- People: are the human resources within an organisation.
- Technology: are the tools, knowledge, and techniques used to transform inputs into outputs.
- Tasks: represent an organisation's mission, purpose, or goal for existing.

Although the adoption of a socio-technical perspective has its origins in much earlier works, socio-technical theory is still considered relevant because of its balanced treatment of people, organisations, technology and contexts [Morris, 2009]. Its continued relevance is echoed by [Eason, 2008], who considers it one of the most successful of all organisational theories, offering a framework which provides a means of understanding the complexity surrounding how people interact, as well as how they utilise tools and technologies to achieve their work tasks. As Eason observes, by focusing on the interdependencies between people, their roles, and the technical artefacts they use to perform their tasks, it is possible to understand an organisation's operational reality. Thus there is an assumption that "peoples' everyday thinking and acting is affected

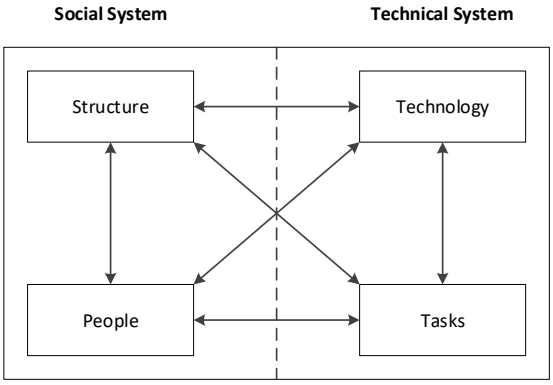


Figure 3.1: Socio-technical theory (adopted from [Bostrom and Heinen, 1977])

by their socio-cultural milieux: by the norms, perceptions, and understandings that they develop or acquire as they interact with others" [Griffith and Dougherty, 2002, p. 212]. Notably, peoples' behaviour with technology is clearly shaped by what occurs within the social domain [Coiera, 2007]. This is certainly evident in terms of V&V. For example, in the context of software testing, the attitudes of senior test engineers can shape the attitudes of junior team members [Shah and Harrold, 2010]; and, in terms of reviews, experienced reviewers are known to resist adapting their adopted techniques when receiving external suggestions [Kelly and Shepard, 2002].

By adopting a socio-technical perspective, we are provided with the concepts necessary to help understand how tasks are performed within an organisation [Eason, 2008]; thereby highlighting the work and those performing it [Griffith and Dougherty, 2002]. As discussed within Chapter 1 (and highlighted within [Kreeger and Harindranath, 2012]): it is essential to understand, within an industrial context, both the technical - and the social - realities of security V&V. Therefore, adopting a socio-technical perspective becomes necessary (especially when considering that "it bridges the gap between social organisations, and the technologies that such organisations use in order to achieve goals" [Morris, 2009, p. 13]). We continue to strengthen the relationship between adopting a socio-technical perspective, and that of V&V, within Section 3.2.

3.2 Software Engineering as a Socio-Technical Activity

Software engineering is a complex social-technical activity [Sawyer, 2004, Henry, 2005, Sjøberg and Grimstad, 2010, Sedano et al., 2017], involving interaction between people (e.g. engineers and managers) and their interactions with specific technical methods and technologies so as to perform their allocated tasks [Sawyer, 2004]. Notably, the

surrounding human issues are regarded as having at least equal importance - or more so - than the technical issues [Tomayko and Hazzan, 2004, Acuña et al., 2006, Fairley, 2009]. Specifically, the social and human aspects of V&V include:

- Interaction: software development, including V&V, is predominately a team effort [Yilmaz and Phillips, 2006] (most modern software products are too complex to be engineered by individuals [Aurum et al., 2002], thereby necessitating teamwork [DeFranco and Laplante, 2017]). For example, test-based activities require tools supporting team communication [Mosley and Posey, 2002]; review-based activities necessitate effective teamwork [Aurum et al., 2002, Tervonen and Harjumaa, 2004]; and, specific security V&V techniques are known to be a combined effort between security engineers and quality assurance engineers [Takanen et al., 2008]. As software development is more sociological than technological in nature, there is greater dependency placed on people communicating and interacting with one another than people communicating and interacting with machines [DeMarco and Lister, 1999]. Notably, a lack of team work is an identified challenge in software development [Ahmed, 2011].

- Collaboration and co-operation: test engineers are known to interact with different types of people, at different levels, both internal and external to their organisation [Hass, 2008]. Therefore, people collaborating and co-operating (e.g. task allocation, the co-ordination of actions and the ability to resolve conflicts) are central activities in software engineering and V&V [Burnstein, 2003, Yilmaz and Phillips, 2006, Martin et al., 2008, Whitehead et al., 2010]. For example, software reviews should always be conducted in the spirit of co-operation [Jawadekar, 2004] and test engineers should focus on defects without attributing blame (this is an identified behavioural challenge) [Everett and McLeod, Jr., 2007]. Conflict surrounding defects can lead to reluctance in developer and test engineer co-operation [Vogel, 2011]. Notably, software testing is an "organizationally cross-cutting activity", involving employees positioned in a variety of roles i.e. they are not just software test specialists [Mäntylä et al., 2012, p. 145].

- Motivation: although aspects of V&V are deemed creative and intellectually challenging [Burnstein, 2003, Myers et al., 2011, Itkonen et al., 2016], there are examples where a lack of motivation has impacted these activities e.g. [Runeson, 2006, Kollanus and Koskinen, 2006]. Notably, software engineers are motivated by conflicting needs [Fairley, 2009], whereas test engineers are motivated by an interest in the task (it must be viewed as an intellectual challenge) and reward and praise [Hass, 2008]. Hass also observes that test engineers must possess

perseverance and endurance. However, a lack of management support is known to cause test engineers to lose both motivation and interest in the task [Perry, 2006]. According to Perry, management should actively motivate test engineers e.g. by stressing the importance of software testing within an organisation and ensuring that the test schedule and the allocated budget are not reduced to compensate for development overruns.

Although V&V encompasses technical tasks, tasks such as software testing also involve consideration of human psychology [Myers et al., 2011], with interpersonal skills being of noted importance [Hass, 2008]. Certainly, software testing includes both technical and managerial problems (an example of the former is determining what constitutes test adequacy, an example of the latter is the estimation of effort to apply to a specific task) [Bertolino, 2003]. It is, therefore, unsurprising that test engineers require technical skills, as well as personal and managerial skills (notably, in the case of the latter, [Burnstein, 2003] states that managers, in order to support software testing within an organisation, need to: establish standards, assess maturity, support cultural changes and assist in test process improvement - see Section 2.5 for further discussion). Further, the training necessary for performing V&V successfully is multifaceted, crossing both the technical and managerial dimensions, with the latter being considered equally important [Balci et al., 2002]. Therefore, it is understandable that human factors play a key role in software development [DeMarco and Lister, 1999, Morisio et al., 2007] and influence V&V, for example, with engineer attitudes affecting unit testing [Ellims et al., 2004] and motivation impacting both test and review-based activities [Runeson, 2006, Kollanus and Koskinen, 2006].

Significantly, such socio-technical issues impact all aspects of software engineering [Henry, 2005]. It is, therefore, important to understand both the social and the technical aspects of V&V to improve existing practice. Whilst this is desirable within any organisational and technological context, we deem the SME context (for the reasons in Section 1.2) and the adopted security perspective (for the reasons in Section 1.3) as having much greater significance to both the producers of software products and the consumers of such products. Certainly, adopting a socio-technical perspective is supported by [DeMarco and Lister, 1999], who assert that the major challenges impacting software development are more sociological than technological in nature. Social factors have a real impact on software quality [Bird et al., 2009], with many software failures being explained by human factors [Hazzan and Tomayko, 2004]. Whilst all engineering disciplines are dependent on people, processes and technology, people are emphasised as the most important factor in software engineering [Fairley, 2009]. Fairley also states that competent people and teams, with motivation, can overcome weak processes and

technology - but, conversely, notes that excellent processes and technology cannot compensate for inadequate skills, lack of motivation or dysfunctional teams. This, coupled with the socio-technical examples given, once again emphasises that a focus solely on the technical side of security V&V would be insufficient.

In summary, the social and technological aspects of software development are deeply interwoven [Messerschmitt and Szyperski, 2003]. As Sawyer indicates: "it is difficult to disentangle the way people do things from the methods, techniques, and computing technologies they use" [Sawyer, 2004, p. 95]. Therefore, to improve productivity, it is necessary to understand both the social and technical aspects of software development [Henry, 2005]. We thus elect to adopt a socio-technical perspective when examining security V&V within UK-based software SMEs.

3.3 Socio-Technical Approaches for Studying Technology

Although many variations exist, within the socio-technical theory field, as to how best to understand the general relationship between the social and technical sub-systems [Aidemark, 2007], we have elected to adopt the Socio-Technical Interaction Network (STIN) framework. However, as STIN is inspired by both Actor-Network Theory and Social Construction of Technology [Meyer, 2006, Meyer, 2007, Reinert, 2009, Shachaf and Rosenbaum, 2009, Urquhart and Currell, 2010, Urquhart and Currell, 2016], we begin by examining these, as well as Sociomateriality, Structuration Theory and Stakeholder Theory.

3.3.1 Actor-Network Theory

The origins of Actor-Network Theory (ANT) are grounded in the work of [Latour and Woolgar, 1986]. As suggested by the name, the premise of an actor-network is that technology develops through alliances between various actors (software is not typically developed by individuals in isolation [Yilmaz and Phillips, 2006, DeFranco and Laplante, 2017]). Such networks are heterogeneous, containing actors from both the social and technical world (notably, software can itself be considered an actor and not just the humans involved in its construction [Baxter, 2000, Tatnall, 2005]). It is the interaction of these actors which builds and stabilises the network. However, where ANT differs to other frameworks, and has received criticism (e.g. [Walsham, 1997, Mwenya and Brown, 2017]), is the blurring of human and non-human actors [Turner, 2005] ([Latour, 1987] employs the term "actant" to cover both). It is through the principles of agnosticism, generalised symmetry and free association that ANT aims to remain impartial towards an actor [Tatnall and Gilding, 1999] ([Callon, 1986] defines

agnosticism as the "impartiality between actors engaged in controversy", generalised symmetry as "the commitment to explain conflicting viewpoints in the same terms" and free association as "the abandonment of all a priori distinctions between the natural and the social").

Central to ANT is the concept of translation. This concerns the development of technology over time [Cressman, 2012] and involves persuading actors that using a technology in a specific way is in their interest i.e. it provides the answer to their problems [Howcroft et al., 2004]. Effectively, stability and social order are continually negotiated [Monteiro, 2001]. Callon defines four moments of translation [Callon, 1986]:

- Problematisation: is where a focal actor (or group of actors [Baxter, 2000]) identifies a problem (or an opportunity [Carroll et al., 2012]) and wins the support of other actors, thereby becoming indispensable to these actors, and thus helping stabilise the network [Callon, 1986, Howcroft et al., 2004]. Baxter, who examined a software development process using ANT, states that the problematisation phase is normally triggered by an individual developer identifying either an improvement to an existing software product, or a new product, and who then has to convince other developers, and management, of the merit of any such proposals [Baxter, 2000].

- Interessement: involves convincing other actors to accept their defined tasks [Callon, 1986, Howcroft et al., 2004, Iyamu and Sehlola, 2012], thus confirming problematisation [Carroll et al., 2012]. Strategies used to establish actor-network indispensability are dependent on context, however, according to [Rhodes, 2009], they can include negotiation, persuasion, seduction, simple bargaining and violence.

- Enrolment: is the forming of a sufficient number of alliances [Howcroft et al., 2004] and occurs on the successful outcome of problematisation and interessement [Rhodes, 2009]. Baxter provides an example of enrolment involving a software project passing its first review gate - this occurs when the formed alliances agree on the outcome [Baxter, 2000].

- Mobilisation: occurs when the actors enrolled within the network are unlikely to withdraw due to the social investment [Howcroft et al., 2004]. The ideas, technology, and the artefacts associated with the actor-network, then become institutionalised [Melian and Mähring, 2008].

In summary, ANT is concerned with how both actors and organisations "mobilise, juxtapose and hold together the bits and pieces out of which they are composed"

[Law, 1992, p. 386] and has been applied to a variety of topics [McLean and Hassard, 2004, Meyer, 2007, Alcadipani and Hassard, 2010, Mwenya and Brown, 2017]. For example, [Baxter, 2000] adopts ANT to help understand why software defects are deemed acceptable; [Melian and Mähring, 2008] consider ANT suitable for examining how Hewlett-Packard adopted open source software development practices; and [Beekhuyzen et al., 2012] employ ANT to understand the motivations behind file sharing, as well as the ideologies of those involved. Beekhuyzen et al. conclude that rich insights can be gained through the adoption of ANT, stating that it acted as a useful guiding theory to explain the practices and motivations of file sharing.

However, ANT has been considered controversial [Walsham, 1997, Alcadipani and Hassard, 2010, Gad and Jensen, 2010, Mwenya and Brown, 2017, Sayes, 2017]. This is primarily due to treating humans and non-humans as equal [Walsham, 1997, McLean and Hassard, 2004, Mwenya and Brown, 2017]; neglecting and suppressing social structures, and considering all actors equal by avoiding micro/macro distinctions [Walsham, 1997, Mitev, 2009, Sayes, 2017] (as Mitev acknowledges, in practice, there are "bigger" actors than others, and treating them the same seems inappropriate - similarly, we feel that developers and test engineers should not be considered the same). In addition, although [Baxter, 2000] views ANT as being reasonably simplistic, [Cressman, 2009] considers it notoriously difficult. This complexity is echoed by others e.g. [Howcroft et al., 2004, Nimmo, 2011]. Further, the differing versions of ANT are known to involve diverse theoretical, methodological and empirical implications [Gad and Jensen, 2010]. We feel that this complexity has contributed towards the tendency of researchers, when employing ANT, to only focus on the prominent and the powerful (as identified within [McLean and Hassard, 2004, Mitev, 2009, Mwenya and Brown, 2017]) which, in turn, is unlikely to result in an unbiased analysis.

3.3.2 Social Construction of Technology

When the phrase "social construction of technology" is broadly used it includes ANT, however, when used more narrowly it refers to a specific approach, namely, SCOT [Bijker, 2010]. SCOT is a sociological approach to the analysis of technological artefacts (which includes machines, technical processes and both hardware and software [Bijker, 1997]) within the context of society [Pinch, 1996]. A social constructivist approach, it rejects technological determinism i.e. that technology develops autonomously and determines societal development to an important degree (thereby providing a linear and one-dimensional view of technological development) [Bijker, 2009, Bijker, 2010]. However, SCOT, and social shaping theorists, have been accused of a form of social determinism [Howcroft et al., 2004] (in contrast, ANT considers technological and social

determinism flawed [Tatnall and Gilding, 1999] and attempts to avoid both [Howcroft et al., 2004] through denying that purely technical, or purely social interactions are possible [Tatnall, 2005]). In turn, SCOT preserves the social environment as it is this which shapes the technical characteristics of the artefact itself [Howcroft et al., 2004]. Howcroft et al. acknowledge that it is crucial to see artefacts as viewed by the social groups themselves, since otherwise technology would be seen as autonomous. Effectively, social constructivists see people, within social groups, as playing a crucial part in technology development [Vermaas et al., 2011].

The SCOT framework consists of the following concepts:

- Relevant social groups: technology development is viewed as a process where multiple groups (each with a specific interpretation) negotiate over its design. These can be institutions, organisations and organised and unorganised groups of individuals [Pinch and Bijker, 1989] (individuals can also span multiple social groups [Bijker, 2010]). These groups are delineated according to similarities in their interpretation of the technology being developed [Howcroft et al., 2004], denoting all members of a social group share the same view of an artefact [Pinch and Bijker, 1989]. However, [Klein and Kleinman, 2002] believe that the assumption, that interactions typically result in consensus, requires rethinking as it overlooks asymmetries in power. It is where social groups differ in their description of an artefact that interpretative flexibility is demonstrated.

- Interpretative flexibility: captures that there is flexibility in how people think of, or interpret, artefacts, as well as flexibility in artefact design i.e. there is not one right way [Pinch and Bijker, 1989, Howcroft et al., 2004]. Notably, when developing software, there are often many solutions to a given problem [Fox, 2006, Kitchin and Dodge, 2011]. Orlikowski considers interpretative flexibility as an "attribute of the relationship between humans and technology", which is, therefore, influenced by the material artefact itself (e.g. the specific hardware and software which comprises the technology), the characteristics of the human agents (e.g. experience and motivation) and the characteristics of the context (e.g. the social relations, task assignment and resource allocation) [Orlikowski, 1992, p. 409]. However, the concept of interpretative flexibility was not without its critics, with [Russell, 1986] considering it insufficient, since: aside from showing what different social groups think about an artefact, it is also necessary to show how they can influence it (i.e. different social groups will differ in their abilities of influence).

- Closure and Stabilisation: with the presence of multiple groups, different in-

terpretations can lead to conflicting interpretations of an artefact [Klein and Kleinman, 2002]. According to Klein and Kleinman, closure is achieved when no further design modifications occur and the artefact stabilises in its final form (either a final decision is reached or no further decisions occur) i.e. the different interpretations are either brought into agreement, or one interpretation becomes dominant [Howcroft et al., 2004]. Effectively, the interpretative flexibility of an artefact diminishes and a common meaning becomes generally accepted [Bijker, 1997, Martin, 1999]. Importantly, closure does not necessarily denote that all problems (i.e. conflicting interpretations) have truly been resolved [Meyer, 2007]. Although within [Bijker, 1997] closure is described as the product of consensus, [Klein and Kleinman, 2002] observe that how this occurs is not detailed - effectively, failing to explain why one group's view was selected and why others were resolved or suppressed. Specifically, they note that more powerful organisations may force closure on others even if the artefact does not work for them. This situation is commonly faced in terms of the resulting communication conflict surrounding defects (e.g. between developers and test engineers, see [Cohen et al., 2004]). Thus there is a need to understand the power and structural relationships between the different social groups. This is particularly important when acknowledging that software development is considered a story filled with compromises [Bashir and Goel, 1999]. Klein and Kleinman also question whether deadlines can force closure [Klein and Kleinman, 2002]. V&V can become "squeezed" towards the end of a project when product release schedules are driven by forces such as being first to market [Nguyen et al., 2006], as well as accounting for delay in other phases of software development [Bashir and Goel, 1999] e.g. software testing has been perceived as a cushion by developers [Murugesan, 1994]. This example reflects both closure by deadline and an asymmetry in power between two social groups (i.e. developers and test engineers). According to [Klein and Kleinman, 2002], closure and consensus can only be understood through examination of the power relationships which exist between groups. Russell considers the failure to appropriately address social structure as a major weakness of SCOT [Russell, 1986].

- Technological frames: refer to the "structure of rules and practices that enable and constrain the interactions among the actors of a relevant social group" [Howcroft et al., 2004, p. 348]. Bijker details a list of elements that comprise a technological frame (e.g. goals, key problems and requirements to be met by problem solutions) [Bijker, 1997], with these elements influencing the interactions and attributing meaning to technical artefacts [Howcroft et al., 2004]. Although

[Klein and Kleinman, 2002] see the introduction of the technological frame as an important first step towards acknowledging structure, they feel more could be achieved. Similarly, [Prell, 2009] views technological frames as a catch-all concept for handling structure; Prell also notes that the heterogeneity of technological frames can mask the more obvious (and potentially most influential) forces at work.

Notably, SCOT has been applied to the study of a variety of technological artefacts e.g. the design of an online information system for use by young people [Prell, 2009] and packaged software selection [Howcroft and Light, 2010]. Each of these studies report differing levels of value, and success, in their adoption. However, whilst [Prell, 2009] believes that SCOT has value, they caution that the resulting rich descriptions can lead to the obscuring of the strongest influences shaping a technology's design. Similarly, [Howcroft and Light, 2010] also view SCOT as a useful approach, but had to augment it to address their needs (in their case, with a political perspective to reflect structural influences; notably, [Meyer, 2007] also felt that exclusive use of SCOT would have resulted in a less rich understanding).

3.3.3 Sociomateriality

Whilst the existing socio-technical literature has actively acknowledged the role of technology, within both society and organisations (see Section 3.1), Sociomateriality was developed to further account for the emergent relationships between the social and the material [Parmiggiani and Mikalsen, 2013] (with the 'social' relating to humans and their structure and the 'material' encompassing non-humans and their material structure [Mueller et al., 2016]). Orlikowski observes that the material (which ranges from the visible e.g. desks, computers and pens, to the less visible e.g. electricity) has often been disregarded, or downplayed, within many existing organisational studies [Orlikowski, 2007]. This, Orlikowski argues, is problematic given the role materiality plays within any form of organisational practice. Therefore, a sociomaterial approach helps emphasise the inherent inseparability between the social and the material i.e. "there is no social that is not also material, and no material that is not also social" [Orlikowski, 2007, p. 1437].

Given its socio-technical basis (the socio-technical approach is considered an important precursor to sociomaterial thinking [Cecez-Kecmanovic et al., 2014]), Sociomateriality has become seen as a popular approach within the information systems field [Kautz and Jensen, 2013, Parmiggiani and Mikalsen, 2013]. Notably, such an approach suggests that software can only be understood when viewed within the context of a so-

cial practice [Riemer and Vehring, 2010]. This is echoed by [Doolin and McLeod, 2012], who view information systems development as a sociomaterial practice. Therefore, and acknowledging the inseparability of work and materiality, [Johri, 2011] adopts a sociomaterial approach when studying the work practices of globally distributed software developers. Similarly, [Riemer and Vehring, 2010], who used Sociomateriality to examine software usability, indicate that it was only by adopting a sociomaterial approach that it was then possible to make sense of, and interpret, their observations.

However, the application of Sociomateriality, within real-world and empirical-based settings, often remains unclear [Weißenfels et al., 2016]. For example, Kautz and Jensen indicate that Orlikowski and Scott (i.e. [Orlikowski, 2007, Orlikowski and Scott, 2008, Scott and Orlikowski, 2009, Orlikowski, 2010]) offer little assistance in terms of understanding how the sociomaterial issues - surrounding information systems - can be studied in practice and thus used to provide new insights [Kautz and Jensen, 2013]. The difficultly of applying Sociomateriality to practice has been echoed by others e.g. [Doolin and McLeod, 2012, Alter, 2012, Mutch, 2013, Mueller et al., 2016, Holeman, 2018]. Significantly, [Mutch, 2013] questions whether Sociomateriality, as propounded by Orlikowski and Scott, is capable of providing insights above that which could be obtained by other approaches, with [Holeman, 2018] stating that it is not uncommon for reviewers of papers to question why Sociomateriality was used over more familiar theories such as ANT. Therefore, whilst many researchers have identified the value of adopting a sociomaterial perspective, [Mueller et al., 2016] states that an equal number of researchers highlight the operational and empirical difficulties associated with such an approach. Notably, Sociomateriality has been considered highly theoretical [Leonardi, 2013] and stands accused of introducing more "academic jargon monoxide" [Kautz and Jensen, 2013, p. 15]. This, combined with the view that researchers adopting Sociomateriality "often find themselves confused regarding research methods" [Elbanna, 2016, p. 84], and reportedly encounter difficulties in terms of data collection and analysis [Elbanna, 2018], may have helped contribute as to why the adoption of a sociomaterial approach is considered to be challenging in application [Weißenfels et al., 2016]. Collectively, this may explain why the five notions associated with Sociomateriality (namely, materiality, inseparability, relationality, performativity and practices) have not been fully engaged with within the existing sociomaterial literature [Jones, 2014].

3.3.4 Structuration Theory

Based on the interconnectedness of structure and agency [Allison and Merali, 2007], Giddens proposed Structuration Theory as a means of examining their relationship [Giddens, 1984]. Notably, Giddens details three dimensions of structure: signification,

domination and legitimation, as well as three dimensions of interaction: communication, power and sanction, with each structural dimension being translated into action by a modality i.e. an agent utilises these structures for social interaction. As Structuration Theory adopts the view that the social actions of humans cannot be fully understood by examining structure or agency alone it, therefore, rejects the dualistic view of structure and agency, in turn, viewing them as being a mutually constitutive duality [Jones and Karsten, 2008]. Thus, with the relationship between individuals and society being the central concern of Structuration Theory [Jones and Karsten, 2008], it is viewed as providing a framework capable of studying the social activities of humans [Giddens, 1991]. As stated within Section 3.2, software engineering, and V&V, are social activities involving humans.

Therefore, there is a history in applying Structuration Theory to the study of information systems e.g. [Rose and Scheepers, 2001, Nicholson and Sahay, 2001, Frederiksen and Rose, 2003, Allison and Merali, 2007, Jones and Karsten, 2008, Allison, 2010, Jones, 2011]. For example, Frederiksen and Rose state that software development can be analysed using Structuration Theory, indicating preference over other approaches (e.g. communities of practice, situated action, and ANT) due to its focus on action and structure [Frederiksen and Rose, 2003]. Nicholson and Sahay draw on Structuration Theory when exploring the political and cultural issues surrounding software outsourcing [Nicholson and Sahay, 2001]; similarly, Allison and Merali employ Structuration Theory when studying software process improvement, observing that its use "explicates the dynamics of emergence in process improvement" [Allison and Merali, 2007, p. 677].

However, Structuration Theory has been regarded as being both complex and abstract [Bryant and Jary, 1991, Rose, 1998, Nicholson and Sahay, 2001, Jones and Karsten, 2008, Jones, 2011]. Significantly, Rose and Scheepers consider it as being "too complex, diverse and alien to be adapted wholesale" [Rose and Scheepers, 2001, p. 219] (Jones and Karsten also observe that only a relatively small amount of information system research has fully engaged with Giddens' work [Jones and Karsten, 2008]). Similarly, the lack of empirical examples found within Giddens' own work, coupled with the complex and abstract nature already referenced, provides the researcher with "few clues as to how to proceed in the everyday world in the gathering of useful understanding, and its reflection back into the world of practice" [Rose, 1998, p. 912]. Rose also states that as information systems is an applied field, there is an assumption that there will be a resulting contribution to practice. We observe that one of the research objectives (see Section 1.4.2), as well as one of the motivating factors behind this thesis (see Section 1.1), is to understand security V&V practice in order to improve upon such

practice. Further, and given our stated intent to perform an empirically-based software engineering study (see Section 4.1), we also observe that there is reportedly limited empirical work which utilises Structuration Theory [Kolarz, 2016]. Additionally, and given that our research focuses on the creation and use of technological artefacts (see Section 3.4), we find, within Structuration Theory itself, that technological artefacts are almost completely overlooked [Jones and Karsten, 2008] (as [Jones, 2011] further expounds, across the extensive works of Giddens, there is little reference to technology or information systems-based phenomena).

3.3.5 Stakeholder Theory

We have already acknowledged that software engineering is a team-effort [Yilmaz and Phillips, 2006, DeFranco and Laplante, 2017] (see Section 3.2), involving a variety of stakeholders [McManus, 2004, Power, 2010a, Power, 2010b]. Therefore, and given the diversity of these stakeholders, coupled with the view that effective information systems development requires their successful participation and interaction [Pan and Flynn, 2003, Pan, 2005], Stakeholder Theory was proposed as a means of enabling managers, within organisations, to both understand, and manage, these stakeholders more strategically [Aaltonen and Kujala, 2016]. Notably, Stakeholder Theory helped challenge the view that stockholders should be the primary beneficiaries of an organisation's activities [Phillips, 1997], thus aiming to create value for all stakeholders [Biesenthal and Wilden, 2014] (with the term 'stakeholder' deliberately contrasting with 'stockholders' or 'shareholders' [Scholl, 2001]).

Whilst Stakeholder Theory is rooted in the strategic management literature [Sharp et al., 1999, Power, 2010a, Power, 2010b, Vrhovec et al., 2015], and is extensively discussed within management-based literature more generally [Niemi, 2007], it has also been applied to other areas [Vrhovec et al., 2015], including within an information systems context [Pouloudi and Reed, 1998, Pan and Flynn, 2003, Pan, 2005]. For example, it has been employed to understand the reasons for abandoning information systems development projects [Pan and Flynn, 2003, Pan, 2005] and it has also been posited as a means for developing models which can help managers understand both who, and what, makes a difference during software development [Power, 2010a, Power, 2010b]. However, in terms of the evolution of Stakeholder Theory itself, we find that: "the impact of the development of information technology and its influence on particular groups of stakeholders has been perceived in quite a limited way" [Wielki, 2011, p. 497].

Notably, most of the approaches to stakeholder analysis - in both a strategic management and information systems context - have failed to provide adequate guidance on stakeholder identification [Pouloudi and Reed, 1998]. The difficultly of stakeholder

identification has been echoed by others e.g. [Phillips, 1997, Sharp et al., 1999, Smith and Hasnas, 1999, McManus, 2004, Watson et al., 2009], with Smith and Hasnas also indicating, as a significant weakness of Stakeholder Theory, that it fails to identify how stakeholder interests should be balanced (it also fails to indicate how stakeholders should be represented and how power should be distributed in order to protect their interests [Cummings and Patel, 2009]). Further, and due to its conceptual breadth, whilst the term 'stakeholder' is considered as being powerful, it is also one with many interpretations [Phillips et al., 2003] and associated definitions [Juhola et al., 2014]. Freeman defines a stakeholder as "any group or individual who can affect or is affected by the achievement of an organization's objectives" [Freeman, 2010, p. 46], however, as Watson et al. question: "where does one accept that someone affects or is affected by said objectives?" [Watson et al., 2009, p. 313]. Collectively, these concerns may help explain why the application of Stakeholder Theory - especially when applied to the study of information systems - is often perceived as being particularly problematic [Smith and Hasnas, 1999] and thus why there is no formal application of Stakeholder Theory in either a software development team or process context [Power, 2010a, Power, 2010b]. Therefore, supporting the view that Stakeholder Theory is "normally confined to academic theories of business ethics" [Barry, 2002, p. 542].

Having examined ANT and SCOT, as well as Sociomateriality, Structuration Theory and Stakeholder Theory, we now justify the adoption of STIN.

3.3.6 Socio-Technical Interaction Networks

A Socio-Technical Interaction Network (STIN) is defined as a "network that includes people (including organizations), equipment, data, diverse resources (money, skill, status), documents and messages, legal arrangements and enforcement mechanisms, and resource flows" [Kling et al., 2003, p. 48] (further, and aside from people and organisations, they can also include their practices and technology [Walker and Creanor, 2009, Shachaf and Rosenbaum, 2009]). All such elements are heterogeneous, with the relationships, between elements, including social, economic and political interactions. It is these structured relationships, between the various elements, which result in a network. Notably, the actors within a network can play multiple roles [Kling et al., 2003] and participate in multiple, overlapping networks [Letch and Carroll, 2007]. In summary, STIN identifies groups of interactors and then models the relationships between them [van der Merwe, 2010], thereby emphasising the "importance of the character of interactions between people, between people and equipment, and even between sets of equipment" [Kling et al., 2003, p. 49].

Based upon this, STIN adopts the view that the technical and the social world

- being highly intertwined - co-constitute one another [Kling et al., 2003]. It is by stressing this inseparability [Urquhart and Currell, 2010], that insight is provided into the way they mutually shape one another [Villar-Onrubia and Rajpal, 2016]. Thus a STIN is viewed as providing a systematic, socio-technical model, accounting for both the social and technical systems and their interactions [Shachaf and Rosenbaum, 2009]. It is this integration which is seen to allow STIN to offer richer explanatory power [Horton et al., 2005]. The view that the social and the technological are not meaningfully separable is considered by Kling et al. as one of the four fundamental assumptions underling the STIN methodology. The other assumptions (detailed within [Kling et al., 2003]) indicate: that theories of social behaviour should influence technical design choices; that participants are involved in multiple, non-technologically mediated social relationships, therefore, having multiple, possibly conflicting, roles; and that focus should centre on routine use as opposed to just that of adoption and innovation (as Orlikowski and Iacono state: "Latour theorizes about how new technologies come to *be*; Kling and Scacchi theorize about how new technologies come to be *used*" [Orlikowski and Iacono, 2001, p. 126]).

STIN develops several concepts from SCOT and ANT into a usable framework [Urquhart and Currell, 2010, Urquhart and Currell, 2016, Bingham-Hall, 2017]), with [Kling et al., 2003] distiling STIN into eight steps (which they indicate are illustrative as opposed to enumerative - this is echoed by [Letch and Carroll, 2007]):

- Identify a relevant population of system interactors: involves identifying the actors, their roles and the way they participate within a system. STIN attempts to understand the characteristics and scope of the interactors ahead of time (as opposed to following them i.e. before undertaking fieldwork [Meyer, 2007]) [Kling et al., 2003]. According to [Shachaf and Rosenbaum, 2009], this should encompass technical and non-technical stakeholders. Software development involves both, with each impacting the construction, and shape, of a software artefact. In addition, adopting the categories of "people, organisations, data, equipment/tools, policies/guides" leads to a more balanced view of human and non-human agents [van der Merwe, 2010, p. 45] (STIN rejects the strong symmetry principle found in ANT whereupon non-humans can possess the same range of actions as humans [Kling et al., 2003]). However, identifying what belongs within a network - and that which does not - is a significant challenge [Kling et al., 2003] (this has also been reported in terms of Stakeholder Theory and stakeholder identification, see Section 3.3.5). Meyer states, as a limitation of STIN, that the ability to identify and analyse STINs is dependent on the researcher acquiring access to, and information from, individuals [Meyer, 2007]. Similar issues have been identified

in terms of SCOT i.e. when identifying relevant social groups [Pinch and Bijker, 1989, Klein and Kleinman, 2002] and ANT [McLean and Hassard, 2004, Mitev, 2009].

- Identify core interactor groups: in this step the identified interactors are grouped together by role (which draws attention to their interactions [Taylor-Smith, 2016]). In the scholarly communication forum example presented in [Kling et al., 2003], relevant interactor groups might include authors, reviewers, readers, grant funders and departmental administrators. Kling et al. also note that the identified groups might overlap and, in some instances, have conflicting roles. Similarly, when modelling open source software processes as a STIN, [Jensen, 2010] observes that participants may have multiple, possibly conflicting, roles. In our study context, developers who also perform a V&V role, have, by virtue of this, conflicting roles (sole reliance on developers testing or reviewing their own code should be discouraged to remove bias [Root and Sweeney, 2006], however, the limited resources of SMEs can lead to individuals performing multiple roles [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018]). Understanding this role conflict is essential in determining the level of independence in V&V activities within SMEs.

- Identify incentives: the incentive structures of the interactors are then identified [Kling et al., 2003], as are their motivations [Meyer, 2007]. Given the impact of motivation on engineers performing V&V (see Section 3.2), and the relationship between the social construct of motivation with the perceived technical success of a product, it is essential to understand an interactors' incentives in achieving software security - through the application of security V&V - as well as understanding the motivating factors for performing the associated activities. However, there is an assumption that their motivations are both relatively transparent - and correctly reported - although, in practice, humans are not always predictable, with their own motivations not always being clear to them [van der Merwe, 2010]. Based upon this, van der Merwe indicates rigor is required during the knowledge elicitation process.

- Identify excluded actors and undesired interactions: aside from focusing on the desired interactions within a network, it is critical to understand the interactions interactors do not wish to have [Kling et al., 2003]. Further, identifying excluded interactors, and undesired interactions, is an important, although often overlooked step within other socio-technical approaches [Meyer, 2007]. Kling et al. provide "entanglement with bureaucracy" as an example of an unwanted in-

teraction, citing that developers of a system may prefer to develop at their own location - as opposed to a customer location - so as to avoid negotiating with the customer's systems administration. Understanding these aspects, in terms of the socio-technical realities surrounding security V&V practice, within SMEs, is essential to improving current practice (i.e. it provides crucial information on the factors impacting and influencing a network [Taylor-Smith, 2016]).

- Identify existing communication forums: this step involves identifying the existing communication systems used by the interactors. In Urquhart and Currell's application of STIN, within a health informatics context, they identify "patient-professional communication" and "interprofessional communication systems" and elect to focus on the places and medium of communication [Urquhart and Currell, 2010]. Thus, in our study context, we can envisage developer-test engineer communication as being one such example, involving not just face-to-face communication (i.e. non-technologically mediated social relationships [Kling et al., 2003, Jensen, 2010]), but also technologically mediated relationships, including communication via email and defect tracking and code review systems. As discussed within Section 3.2, communication is a central concept in V&V.

- Identify resource flows: involves understanding how resources flow throughout the network. Resource flows can have direct and indirect influence on interactions within the network and, according to [Kling et al., 2003], this involves "following the money". Notably, resource flows can also include power, political interests and special interest groups [Letch and Carroll, 2007], as well as expertise [Meyer, 2007, Taylor-Smith, 2016]. All of these factors are clearly relevant to software development and V&V.

- Identify system architectural choice points: a choice point is defined as a "technological feature or social arrangement in which the designer can select alternatives" [Kling et al., 2003, p. 58]. During the process of software development there are many architectural choice points that are encountered which require decisions to be made which select one way over another [Meyer et al., 2011]. This was discussed, in the context of SCOT, under the concept of interpretative flexibility (STIN is considered to be consistent with SCOT in this regard [Meyer, 2007], see Section 3.3.2).

- Map architectural choice points to socio-technical characteristics: having identified the socio-technical features of the interactors, as well as the architectural design choice points, it is then possible for the analyst to develop an understand-

ing of which "combinations and configurations of features are most compatible, viable, and sustainable" [Kling et al., 2003, p. 58]. As Kling et al. observe, this may involve identifying tradeoffs (software projects generally involve tradeoffs between features, reliability, time and money [Kaner et al., 2002]).

However, and given the steps above, we observe that there is discussion on what STIN actually is, for example: is STIN a methodology [Kling et al., 2003], an emerging conceptual framework [Scacchi, 2005], a modelling approach [Letch and Carroll, 2007], a conceptual model and framework [Shachaf and Rosenbaum, 2009], an analytic strategy (as opposed to a body of theory) [Meyer, 2007, Walker and Creanor, 2009], a framework for a method of inquiry [Urquhart and Currell, 2010, Urquhart and Currell, 2016], an analytical framework [Villar-Onrubia and Rajpal, 2016, Taylor-Smith, 2016] or a theoretical framework [Barab et al., 2003, Bingham-Hall, 2017]. Further, although STIN has not yet become widely adopted [Meyer, 2007], interest in its adoption is developing [Walker and Creanor, 2009]. This has led to some considering it as having been undervalued [Bingham-Hall, 2017], with [Urquhart and Currell, 2010] observing that STIN has been employed within a variety of complex settings to understand the use of information technology. For example, Urquhart and Currell, who applied STIN to understand and explain controversy around home uterine monitoring, concluded that it provided a useful perspective, as well as value beyond that of just conducting a systematic review. However, they also noted that some adaptations might be necessary to address aspects of their particular focus i.e. the clinical usage of information technology. Notably, although the focus in [Kling et al., 2003] was upon scientific collaboratories, [Letch and Carroll, 2007] observe that there were broader applications, as Kling et al. state: "STIN models help us to understand human behaviors in the use of technology-mediated social settings" [Kling et al., 2003, p. 48]. We would argue that the process of software development, and therefore, the activity of security V&V, fits perfectly within the definition of a technology-mediated social setting (STINs can also contain non-technologically mediated social relationships [Kling et al., 2003, Jensen, 2010]). Based upon this, [Letch and Carroll, 2007] apply STIN to study the effects of e-government on marginalised people, aggregating some of the steps and also extending it by classifying stakeholders according to their levels of interest and influence. We feel this is a useful extension, which helps lead to an understanding of some of the structural aspects that will exist within a network (such structural aspects were discussed within the analysis of SCOT, see Section 3.3.2). However, van der Merwe considers STIN as a methodology which: "surfaces relationships, intentions and motivations, in particular power and influence in networks" [van der Merwe, 2010, p. 21]. Thus, van der Merwe views STIN as a productive lens, which reportedly helped make sense, as

well as present, aspects of human behaviour when analysing an existing system (in this instance, the design-orientated stages of STIN were de-emphasised).

More pertinent to this thesis, [Scacchi, 2005] applied STIN to the study of free/open source software development processes in order to explore the social and technological interactions found within different research and development communities. Scacchi acknowledges that STIN draws attention to the "relationships that interlink what people do in the course of their system development work to the resources they engage and to the products (software components, development artifacts, and documents) they create, manipulate, and sustain" [Scacchi, 2005, p. 3]. Scacchi concludes that STINs provide a better way to understand software development (including understanding why and how software developers participate in free/open source projects, as well as what sustains their interest). Similarly, when applying STIN to the study of open source software processes, [Jensen, 2010] states, when comparing STIN with other socio-technical models, that STIN provides a richer, and a more complete understanding of human behaviour.

In summary, network models represent the complexity in the relationships between people and technology and STIN is one example of a network model [Walker and Creanor, 2009] ([Mostashari, 2010] views STIN as one of the main analysis methodologies for socio-technical networks). However, although there is clear promise associated with STIN, there is, as Meyer indicates, a need to more fully explain the methods and tools a researcher should use when embracing STIN [Meyer, 2006, Meyer, 2007]. Meyer also adds that its nebulosity may lead to it being overlooked by new researchers and graduate students. That there is little guidance, beyond the eight steps captured within [Kling et al., 2003], has been acknowledged by others e.g. [van der Merwe, 2010]. However, there is a growing body of research which utilises STIN [Rosenbaum and Joung, 2004]. Also, as with other approaches, adaptations, in some applications, have been seen as necessary to the original eight steps proposed by Kling et al. e.g. [Rosenbaum and Joung, 2004, Letch and Carroll, 2007, Urquhart and Currell, 2010], with van der Merwe broadly following these steps [van der Merwe, 2010]. However, [Reinert, 2009] indicates that STIN affords a more nuanced view than other socio-technical approaches, with [Kling et al., 2003] stating that standard models can ignore the complexity surrounding human motivation, as well as their relationships with others. We observe that modelling such aspects, which have a clear impact on actual security V&V practice (see Section 3.2), is essential in understanding the surrounding socio-technical realities of such practice. Scacchi utilised STIN to examine networks of people who work together to create complex software products [Scacchi, 2005]. Further, it is considered a tool for understanding socio-technical systems that privileges neither the social or technical [Meyer, 2006, Meyer, 2007], with it highlighting the relationships between people and

technologies, as well as their mediating practices [Walker, 2008]. We believe that modelling security V&V, as a Socio-Technical Interaction Network, will suitably highlight the socio-technical realities which surround this complex socio-technical activity. We provide further justification within Section 3.4.

3.4 Justification and Application of STIN to V&V

Socio-technical research focuses on both social and technical interactions that occur in such a way that it is difficult to separate them [Tatnall, 2005]. As observed within Section 3.2, software engineering, and V&V, involve such a combination of interactions. It is, therefore, prudent to adopt a framework capable of accurately representing the richness of these interactions. Having justified that software engineering, and V&V, are socio-technical activities, we now justify the adoption of STIN.

It is through reviewing the steps of STIN (see Section 3.3.6) that we develop a conceptual model (Figure 3.2) to show how STIN can be applied when studying security V&V practice:

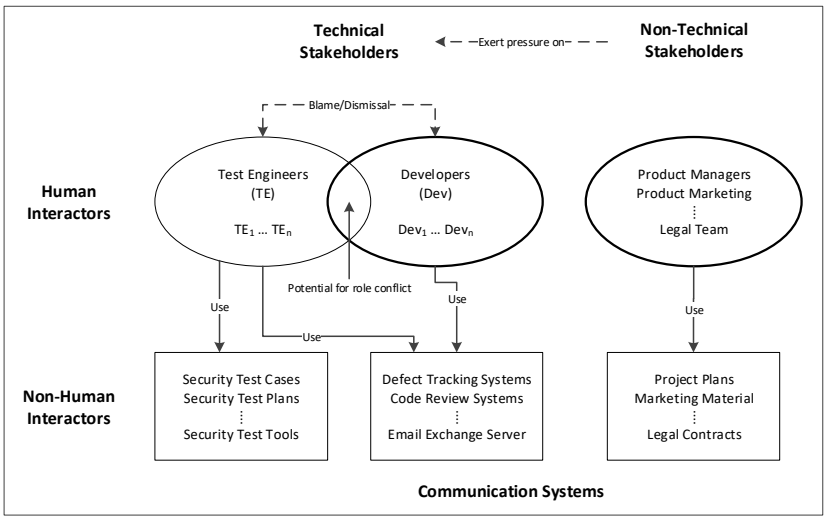


Figure 3.2: Conceptualising V&V as a Socio-Technical Interaction Network

- Step 1 - Identify a relevant population of system interactors: we have already observed that software development involves a combination of technical and non-technical interactors (examples of the former include developers and test engineers, an example of the latter includes product managers). Aside from human agents, software development also includes non-human agents which, from a V&V perspective, includes tools and processes - which need to be included in an analysis

as they influence, and shape, the activity of V&V (see Sections 2.4 and 2.5). Figure 3.2 shows some of the interactors directly and indirectly involved with V&V and provides a simplified view as to why adopting STIN is suitable for the study of V&V. As seen within Figure 3.2, we distinguish between human and non-human agents by ellipses and rectangles respectively ([Walker, 2008] considers this as one of the significant differences between ANT and STIN i.e. the rejection of ANT's strong symmetry principle). Further, we divide these agents into technical and non-technical stakeholders (as suggested by [Shachaf and Rosenbaum, 2009]). It is important to capture this diversity - and to not just focus on the technical (as many software development studies do e.g. [Rooksby et al., 2009], see Chapter 2) - to ensure that the reality of security V&V practice is understood.

- Step 2 - Identify core interactor groups: we have previously noted that interactor groups might overlap, and that role conflict can potentially occur within SMEs due to employees wearing multiple hats [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018]. Of course, coupled with conflict, there is also group power to consider. For example, the literature (although the situation within an SME context is unclear) suggests that developers have greater power than test engineers e.g. [Cohen et al., 2004]. Figure 3.2 shows how interactors can be grouped together (grouping is represented by the ellipses and rectangles) and includes an example of where role conflict could occur when groups overlap (notably, when developers have to perform a software testing role).

- Step 3 - Identify incentives: it is essential to understand the interactors' incentives, as well as their motivating factors, for performing security V&V. In terms of Figure 3.2, this would include individuals, groups and, most importantly, those which find themselves in overlapping groups. It is also pertinent to understand the motivation, or lack of motivation, if applicable, for not performing security V&V. We recall that motivation plays a significant role in software development (see Section 3.2) and that software engineers are motivated by various factors e.g. task identification and trust [Beecham et al., 2008, Hall et al., 2008]. We do not limit ourselves during the process of data collection by excluding any of these factors, however, we acknowledge that the data collected may lead to some factors receiving greater treatment than others (see Section 8.4 for further discussion).

- Step 4 - Identify excluded actors and undesired interactions: Figure 3.2 shows examples of potentially undesired interactions (represented as arrows with dashed lines) e.g. test engineers approaching defect communication with an attitude of blame, developers adopting a dismissive attitude towards test engineers and non-

technical stakeholders applying pressure to deliver a product before testing is complete. All of these are real-world examples and, as with identifying the factors which influence interactor motivation, we do not explicitly exclude any factor which could result in undesired interactions e.g. interactors demonstrating a resistance to change in terms of process improvement initiatives. As noted by [Meyer, 2007], although understanding these aspects is important, other types of socio-technical research often overlook these (notably, within Structuration Theory, Giddens adopts a positive, empowering view of human agency that is voluntaristic in nature [Greener, 2008, Jones and Karsten, 2008, Jones, 2011], however, such a voluntaristic view has been deemed inappropriate [Jones, 2011] - running counter to the assumptions held within the information systems field [Jones and Karsten, 2008] - with Giddens being seen as "gloss[ing] over the impact of power relations in organizations" [McPhee et al., 2014, p. 92]). This is echoed by [Morrison, 2014], who highlights that undesired interactions can significantly reduce motivation, with [Taylor-Smith, 2016] acknowledging the importance of identifying excluded interactors. Capturing this type of information is essential for achieving a balanced view of security V&V practice as found within SMEs.

- Step 5 - Identify existing communication forums: software development involves various types of communication systems, some of which are generic e.g. email, wiki, and version control systems (so as to communicate, discuss and store designs, code and test results for example). However, some are specific to V&V, such as defect tracking systems (where issues with the software, and the supporting artefacts, are raised and discussed between the various stakeholders) and code review systems (where an implementation can be critiqued). See Figure 3.2 for where such communication systems would sit within a V&V STIN. As software is not typically developed in isolation by an individual (i.e. it is a team effort [Yilmaz and Phillips, 2006, DeFranco and Laplante, 2017]), it is necessary to establish how technical and non-technical stakeholders communicate between themselves and each other, as well as how human and non-human agents interact. The importance of communication, within software organisations, is detailed within Section 3.2.

- Step 6 - Identify resource flows: this includes factors such as money, expertise, power and political interests, all of which influence, and shape, security V&V. For example, although there is some debate on the value in conforming to FIPS 140-2 [NIST, 2001], from a security perspective (e.g. see [Deeg and Schreiber, 2009]), it is viewed as a means of widening the customer market available to a software

organisation (as certain consumers demand conformance to this standard). This is an example of where external pressure to conform influences the security V&V being performed within an organisation. However, as in most organisational types and contexts, there will exist asymmetries in power between both individuals and groups (within Figure 3.2, groups with a thicker line possess greater power) - this is certainly the case within software producing organisations.

- Step 7 - Identify system architectural choice points and Step 8 - Map architectural choice points to socio-technical characteristics: these two remaining steps do not relate as well in terms of addressing the research objectives of this thesis. Specifically, as defined in [Kling et al., 2003], a choice point is a technological feature, or a social arrangement, from which a designer can select alternatives (they provide the example of selecting between a pull-based web-board or a push-based mailing list). This involves exploring the past and potential futures [Taylor-Smith, 2016, Taylor-Smith and Smith, 2018]. Notably, we are interested in examining the current state of V&V practice within software SMEs. Therefore, we elect to exclude these more design-focused steps (which is in keeping with the spirit of advice given by Kling et al. on the adoption of STIN and has been followed by others e.g. [Letch and Carroll, 2007, van der Merwe, 2010]).

STIN has been viewed as being built upon, or drawing influence from, ANT and SCOT [Rosenbaum and Joung, 2004, Meyer, 2006, Meyer, 2007, Reinert, 2009, Shachaf and Rosenbaum, 2009, Urquhart and Currell, 2010, Urquhart and Currell, 2016], therefore, it was pertinent to examine each of these (see Sections 3.3.1 and 3.3.2). Reinert notes that STIN aims to address some of the weaknesses of ANT and SCOT in order to provide more complete explanations [Reinert, 2009]. Specifically, Reinert uses STIN to study an organisation's business model and information system and concludes that a more nuanced view was afforded by its adoption. Rosenbaum and Joung, who also state that STIN draws on SCOT, acknowledge the usefulness of STIN across a variety of domains and adopt STIN for studying a digital library, asserting that STIN itself "allows for a fine-grained analysis of the complex relationships among the various components of the socio-technical networks within which ICTs [Information and Communication Technologies] are designed, implemented, and used" [Rosenbaum and Joung, 2004, p. 207].

We also observe that others, when confronted with a choice between ANT, SCOT and STIN, have elected to adopt STIN. For example, Meyer states that whilst ANT, SCOT and STIN are related, they are not identical, however, all provide insight into the role social behaviour plays in the creation and use of technological artefacts [Meyer,

2006, Meyer, 2007] (in contrast, we note that both Giddens and Structuration Theory overlook the technological artefact [Jones and Karsten, 2008, Jones, 2011]). V&V is a fundamental aspect of software development, and utilises various tools (the use of technological artefacts) in which to create a software product (the creation of a technological artefact). It is also noted that they all reject technological determinism. Meyer adds that STIN is more prescriptive than either ANT or SCOT, which is something that [Reinert, 2009] viewed as being of assistance (in contrast to ANT's inaccessibility [Howcroft et al., 2004], and the highly theoretical nature of Sociomateriality [Leonardi, 2013], STIN is not considered an "esoteric social theory" [Kling et al., 2003, p. 56]). Notably, Meyer adopts STIN and states that it focuses more on patterns of routine use than adoption or innovation (again, in contrast, [Prell, 2009] indicates that SCOT's technological frames can mask the more obvious forces at play, which may also be the most influential). As security V&V practice, within UK-based software SMEs, is effectively an unknown [Kreeger and Harindranath, 2012], understanding patterns of routine use has obvious value (especially since people within SMEs "are always busy with their daily routines" [Wong and Aspinwall, 2004, p. 54]). In addition, we note that [Scacchi, 2005] applied STIN to the study of software development, therefore, applying STIN to the study of security V&V could be viewed as a logical, more focused continuation of Scacchi's work. Similarly, Jensen offers further support by indicating that "software development work tasks (i.e. work processes and practices) can be viewed as intrinsically linked networks of interaction between people, tools, and development artifacts, or socio-technical interaction networks (STINs)" [Jensen, 2010, p. 48].

3.5 Conclusion

It is necessary to adopt a socio-technical approach when studying security V&V within software SMEs e.g. test-based activities are socio-technical activities [Martin et al., 2007] impacted by social and organisational problems [Rooksby et al., 2009]. Organisations are, of course, a collection of individuals and groups of individuals [Fairley, 2009]. These individuals need to interact with one another: "[s]uccessful software development requires open communications - within the project group, between the programmers and the users, and with management" [Rob, 2003, p. 95]. Although this listing is incomplete (notably, overlooking test engineers), it reinforces the fact that any framework must identify the different groups and their interactions to be meaningful. This is captured within STIN by the identification of the interactors and their groups (Steps 1 and 2) and the mechanisms by which they communicate (Step 5). In addition, as software development involves individuals interacting within an organisational

structure, it is critical to understand the interrelations which occur between the technical, the product-oriented activities, and the organisational factors which affect the performance of software processes [Yilmaz and Phillips, 2006]. As noted, identifying the incentives (Step 3), the excluded actors and the undesired interactions (Step 4), and the resource flows (Step 6), will help understand how social factors can influence V&V. This is essential given their impact on software testing [Shah and Harrold, 2010]. Shah and Harold conclude that ethnographical studies, focusing on the human and social aspects of software testing, are important as they provide more profound insights into actual practice. Adopting a socio-technical perspective contributes two-fold to understanding practice: by highlighting the work and those performing it [Griffith and Dougherty, 2002]. Currently, neither of these points are understood in terms of security V&V as practiced within UK-based software SMEs. In summary, STIN provides a socio-technical framework that will not only help provide this understanding, but is also a framework that applies well to the study of security V&V.

Chapter 4

Research Methods and Approach

Within this chapter we justify the approach, and the research methods adopted, to address the research question and the associated research objectives. We then detail how the survey instrument and the interview guide were developed. Specifically, this chapter provides:

Section 4.1: Justification for studying aspects of software engineering empirically.

Section 4.2: The research methods and approach adopted.

Section 4.3: The use of surveys in the first phase of data collection.

Section 4.4: The use of interviews in the second phase of data collection.

Section 4.5: The research objectives are examined in light of the selected research methods, leading to the development of a survey instrument and interview guide.

Section 4.6: The adopted research methods and approach are justified.

We begin by highlighting the need for studying software engineering empirically.

4.1 Empirical Software Engineering

Empirical studies allow researchers to understand "how and why things work", thereby permitting a positive affect to be had on software development practice [Perry et al., 2000, p. 347]. Whilst interest in studying software engineering empirically continues to grow [Seaman, 1999, Sim et al., 2001, Singer and Vinson, 2002], it reportedly took many years to gather both momentum and direction [Jeffery and Votta, 1999, Weyuker, 2011] (which is in contrast with other disciplines [Ciolkowski et al., 2003]). Thus it was rare for software engineering research papers (with software testing cited as a specific example) to contain any form of empirical evidence [Weyuker, 2011]. However, by addressing the human aspects, studying software engineering empirically became viewed as having grown in maturity [Seaman, 1999]. As acknowledged by Seaman, this added complexity to an already complex area of research.

As software development is a human intensive activity (see Section 3.2), empirical methods are essential for its study [Wohlin et al., 2003]. Notably, by providing the researcher with the opportunity to work with people [Jeffery and Votta, 1999], it is possible to deal with both multidisciplinary and interdisciplinary factors, such as: issues regarding difficulties in communication, the quality of processes, as well as a myriad of other socio-technical issues involving humans [Harrison et al., 1999]. Therefore, we justify performing an empirical study as follows:

- Empirical methods can be used to investigate software engineering processes and practice [Harrison et al., 1999, Perry et al., 2000, Singer and Vinson, 2002, Tichy and Padberg, 2007]. This reflects the essence of this research i.e. to understand security V&V practice within UK-based software SMEs (see Section 1.4.2).

- It enables the researcher to include human and social elements [Jeffery and Votta, 1999, Harrison et al., 1999, Wohlin et al., 2003]. Within Section 3.2 we observed the importance of studying V&V as a socio-technical activity. Therefore, performing an empirical study complements the adoption of STIN (see Sections 3.3.6, 3.4 and 3.5).

We now expound upon the methods and approach adopted.

4.2 Determining the Methods and Approach

Several types of empirical study exist [Jeffery and Votta, 1999, Perry et al., 2000, Sillitti, 2009], each appropriate for addressing different types of questions and contexts [Sillitti, 2009]. Determining which to employ is considered complex [Jeffery and Votta,

1999], with the resources available to a researcher also influencing the approach [Singer et al., 2008]. As empirical studies are both expensive and time-consuming [Perry et al., 2000], it is important to understand the advantages and disadvantages of each [Sim et al., 2001]. However, before determining which methods to employ, we observe that empirical studies can provide both quantitative and qualitative information [Harrison et al., 1999]. Thus we heed the advice of Harrison et al., namely, that whilst quantitative methods are important in empirical software engineering studies, qualitative methods should not be neglected. Indeed, there is a noted move away from solely quantitative studies [Basili, 2006], with qualitative research methods being necessary to study the complexities inherent in human behaviour e.g. motivation and communication [Seaman, 1999]. As observed within Section 3.2, these social constructs influence both test and review-based activities (e.g. [Kelly and Shepard, 2002, Shah and Harrold, 2010]).

STIN explicitly focuses on aspects such as motivation and communication. Therefore, the richness of the data which can be obtained when adopting STIN indicates a mixed methods approach. Fundamental to such an approach is the combination of quantitative and qualitative research methods within a single study [Creswell and Plano Clark, 2011, Ågerfalk, 2013, Venkatesh et al., 2013, Venkatesh et al., 2016]. Viewed as the third methodological movement, this rejects the traditional quantitative and qualitative dichotomy [Tashakkori and Teddlie, 2010]. Underlying such a movement is the philosophical worldview of pragmatism [Johnson and Onwuegbuzie, 2004].

4.2.1 Pragmatists and Pragmatism

There are several philosophical worldviews, or paradigms, that exist and which help guide the actions of researchers. As [Mingers, 2001] highlights, the existing literature acknowledges at least two, primary worldviews: positivism and interpretivism. Each of these worldviews has a set of ontological, epistemological and methodological beliefs associated with its adoption. In summary, positivists (and postpositivists) believe in a singular reality which can be measured objectively i.e. reality is independent to the researcher; in contrast, interpretivists (and constructivists), believe in multiple, existing realities which are socially constructed by how humans see, and interpret, their reality i.e. research becomes a subjective process due to the inseparability of the researcher and reality. Thus we find that positivists, methodologically, are more closely aligned with quantitative-based methods (e.g. surveys); whereas, interpretivists, are more closely associated with qualitative-based methods (e.g. interviews). Therefore, and given their opposing natures - in terms of their ontological, epistemological and methodological positions - it is unsurprising that there exists considerable debate surrounding their adoption [Fitzgerald and Howcroft, 1998]. Notably, the ensuing ‘paradigm wars’ have

resulted in a focus on their differences, rather than their similarities [Onwuegbuzie and Leech, 2005]. However, as [Weber, 2004] states, much of the rhetoric concerning positivism and interpretivism is unhelpful and specious in nature.

Significantly, purists believe that such paradigms are mutually exclusive - both from an ontological and epistemological perspective [Rossman and Wilson, 1985] (this resulted in the development of the incompatibility thesis [Howe, 1988], which views the quantitative and qualitative paradigms as being fundamentally incompatible). However, in contrast, pragmatists - who aim to put aside the associated, underlying philosophies - believe that this presents a false dichotomy [Onwuegbuzie and Leech, 2005, Glogowska, 2011]. Notably, pragmatism aims to find a middle ground, thereby helping achieve balance between subjectivity and objectivity [Doyle et al., 2016] (pragmatists view the seeking of epistemological purity an unhelpful aim in terms of performing research [Greene et al., 1989]). Pragmatists, in contrast to positivists and interpretivists, believe in the existence of both singular and multiple realities and, from an epistemological perspective, prioritise addressing the research question [Creswell and Plano Clark, 2011, Venkatesh et al., 2013]. Therefore, rather than aiming to achieve some form of philosophical alignment [Glogowska, 2011], pragmatists are viewed as not being committed to any philosophical worldview or reality [Petersen and Gencel, 2013]. Thus pragmatism "opens the door to multiple methods, different worldviews, and different assumptions, as well as different forms of data collection and analysis" [Creswell and Creswell, 2018, p. 11]. As captured by Creswell and Creswell, this provides a philosophical basis which is grounded in real-world practice (pragmatists value knowledge according to how useful it is for addressing practical problems, hence they value practical knowledge over more abstract knowledge [Easterbrook et al., 2008]).

Therefore, and given its real-world, practice-oriented nature [Creswell and Creswell, 2018], it is perhaps unsurprising to find that pragmatism has become recognised as the predominate worldview within an empirical software engineering context [Petersen and Gencel, 2013]. For example, Russo et al. adopt pragmatism when examining software quality concerns within the Italian IT banking sector [Russo et al., 2017, Russo et al., 2018]. Similarly, Rashid et al. embrace the philosophical position of pragmatism when exploring knowledge retention within open source software communities [Rashid et al., 2018]. Further, we also acknowledge the influence pragmatism has had on information systems research more generally [Goldkuhl, 2012]. These examples, and the fact that pragmatism is seen as adopting an 'engineering approach' to performing research [Easterbrook et al., 2008], helps further justify its adoption when addressing the thesis research question and the associated research objectives.

As captured by [Venkatesh et al., 2013], pragmatists select their research methods

based on how they fit the underlying research question. Therefore, and acknowledging the pragmatic and result-oriented views held by the software engineering research community - especially in terms of research methodology [Runeson and Höst, 2009] - as well as the strong association between pragmatism and mixed methods [Johnson and Onwuegbuzie, 2004, Easterbrook et al., 2008, Morgan, 2014], we elect to adopt a mixed methods approach to data collection and analysis.

4.2.2 A Mixed Methods Approach

A mixed methods approach enables a more comprehensive and complete account to be developed, providing additional corroboration in terms of findings (completeness [Bryman, 2006] and triangulation [Greene et al., 1989, Bryman, 2006]). The reliance on a single type of data (i.e. numbers or words alone) is limiting [Johnson and Onwuegbuzie, 2004, Onwuegbuzie and Leech, 2005, Creswell and Plano Clark, 2011]. In an empirical software engineering context, we observe that by collecting different types of data, through different research methods, that a more complete picture is likely to form [Wood et al., 1999]. Further, both quantitative and qualitative research methods can be utilised to explain the findings generated from the other, as well as helping identify the sample and assisting in instrument construction for the next phase of study (complementarity, development [Greene et al., 1989], explanation and sampling [Bryman, 2006]). A quantitative sample can be used to identify relevant groups for further in-depth study [Brannen, 2005].

Such an approach also enhances the credibility of the findings, with an improved perception of usefulness to practitioners (credibility and utility [Bryman, 2006]). Having observed the divide between V&V theory and practice (see Section 2.6), we recall that one of the specific aims of this thesis is to examine security V&V as practiced within UK-based software SMEs (see Section 1.4.2), thereby enabling such practice to be improved (notably, the adoption of pragmatism complements the understanding of such practice, see Section 4.2.1). Additionally, both quantitative and qualitative research methods have their own associated strengths and weaknesses, although, when employed together, the weaknesses should be removed (offset [Bryman, 2006]). It is by employing both that a researcher can benefit from the strengths of each methodology [Jick, 1979, Johnson and Onwuegbuzie, 2004, Onwuegbuzie and Leech, 2005, Ivankova et al., 2006, Venkatesh et al., 2016].

However, two of the most compelling reasons for employing a mixed methods approach (which also influences the design adopted), is that the collected qualitative data can be used to put "meat on the bones" [Bryman, 2006, p. 106], as well as building upon, and enhancing, the acquired quantitative data (illustration and enhancement

[Bryman, 2006]). We also agree, in terms of practical application, that mixed methods is an intuitive approach to research [Creswell and Plano Clark, 2011], as well as that "researchers who ascribe to epistemological purity disregard the fact that research methodologies are merely tools that are designed to aid our understanding of the world" [Onwuegbuzie and Leech, 2005, p. 377]. According to Onwuegbuzie and Leech, such a polarised view is inappropriate, unreliable and does not represent best practice.

4.2.3 A Sequential Explanatory Design

Having justified performing a mixed methods study in general, we now outline the mixed methods design we intend to utilise. Several options exist, each requiring the researcher to consider the following four items with regards to both the quantitative and qualitative strands (where a strand includes the questions, data collection, analysis and result interpretation): the level of interaction between the strands, the relative priority of the strands, the timing of the strands and the procedure for mixing the strands [Creswell and Plano Clark, 2011].

Based on the above, we adopt a sequential explanatory design (the majority of mixed methods designs utilise a sequential approach [Ivankova et al., 2006, Creswell and Plano Clark, 2011]). This implies a study with three distinct phases: firstly, a quantitative data collection and analysis phase; secondly, a qualitative data collection and analysis phase which builds upon the first phase; thirdly, an interpretation phase where the "researcher interprets to what extent and in what ways the qualitative results explain and add insight into the quantitative results and what overall is learned in response to the study's purpose" [Creswell and Plano Clark, 2011, p. 83]. The primary purpose of such a design, according to Creswell and Plano Clark, is to explain the quantitative results obtained in greater depth i.e. the quantitative data permits a general understanding to develop, whereas the qualitative data refines and explains the quantitative data in more depth [Ivankova et al., 2006]. Such an approach implies that the quantitative and qualitative strands are mixed during the point of data collection using a connecting strategy; whereupon, the results from the first strand influence the data collection within the second strand [Creswell and Plano Clark, 2011]. Elaborating further, Creswell and Plano Clark indicate that the quantitative results should help develop the qualitative strand, for example, by helping define and refine the research questions and enabling purposeful sampling. Notably, a sequential-based approach has successfully been employed within software, and V&V-related, empirical studies e.g. [Storey et al., 2008, Chung et al., 2010, Tahir and Ahmad, 2010, Larusdottir et al., 2010, Rungi and Matulevičius, 2013, Deak and Stålhane, 2013, Pham et al., 2017, Zheng et al., 2017].

4.2.4 Summary of the Methods and Approach

Empirical software engineering studies, conducted within industry, often involve obtaining both quantitative and qualitative data [Wohlin, 2013]. Indeed, it is best to utilise a combination of quantitative and qualitative methods [Seaman, 1999] (where, as Seaman acknowledges, qualitative data can be used to illuminate and "go beyond the statistics"). Further, it is necessary to adopt an approach that enables research-in-the-large (i.e. the ability to develop a broad understanding across large groups of interest) [Kitchenham et al., 1995], as well as offering the ability to determine the current situation (i.e. to conduct research-in-the-past [Wohlin et al., 2003]). Based upon this, we employ a three-phased, sequential mixed methods approach to data collection and presentation.

As captured by Figure 4.1, an online survey (constructed around the key themes within STIN, see Appendix D) is employed during the first phase. Notably, a survey is the single method suggested by [Conradi and Wang, 2003, Tichy and Padberg, 2007, Easterbrook et al., 2008] as being relevant for studying software engineering empirically that we have not discounted (we discount controlled experiments as they are typically used to compare techniques and methods [Jeffery and Votta, 1999, Wohlin et al., 2003], and case studies, as they focus on a specific project or organisation [Wohlin et al., 2003, Höfer and Tichy, 2007] leading to results considered harder to interpret and more difficult to generalise and compare [Zelkowitz and Wallace, 1998, Wohlin et al., 2003, Ciolkowski et al., 2003]). Further, surveys can be used to investigate software engineering processes and products [Singer and Vinson, 2002], therefore, seeming appropriate to address the overarching research question of this thesis.

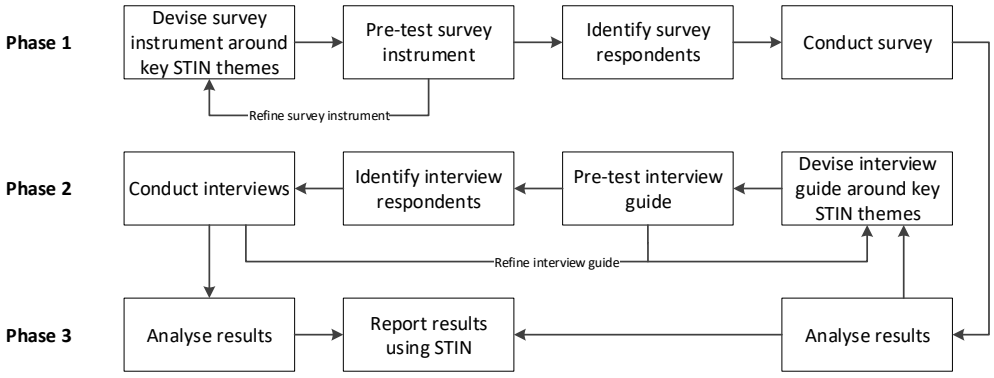


Figure 4.1: Research approach

Within the second phase we conduct interviews with a subset of the respondents from the first phase to develop a richer understanding of the socio-technical realities

faced (the survey thus providing a sampling frame for the interviews [Brannen, 2005]). This second phase is necessary to acquire information about phenomena that cannot be obtained through use of a survey alone (with the questions, within the interview guide, also being developed around the key themes within STIN). Interviews are a common approach to acquiring qualitative data and are useful for collecting information on how software engineering is actually performed [Seaman, 1999]. Finally, within the third phase, STIN is used to structure and present the data acquired through the two previous phases (see [Meyer, 2007] for an example).

In summary, surveys and interviews have been used extensively within empirical software engineering research (see Chapter 2). They have also been employed for understanding information security culture (see Section 4.5.3) and utilised in STIN-based studies e.g. [Meyer, 2007, Kim, 2008, Reinert, 2009, Suri, 2013, Taylor-Smith, 2016, McCoy and Rosenbaum, 2017, Bingham-Hall, 2017]. Therefore, we now examine, and further justify, their use (see Sections 4.3 and 4.4 respectively).

4.3 Data Collection Phase One: A Survey

A survey involves "asking a sample of people from a population a set of questions and using the answers to describe that population" [Fowler, Jr., 2009, p. ix]. It is then possible to generalise the beliefs and opinions of the sample across the population itself [Kasunic, 2005, Easterbrook et al., 2008]. As V&V is a socio-technical activity (see Chapter 3), it is essential to understand the inherent social aspects such as attitudes and motivation: surveys enable the collection of such information [Pfleeger and Kitchenham, 2001]. Notably, it is only possible to reach an understanding about the behaviours and situations of people by asking a sample of people about themselves [Fowler, Jr., 2009].

As the most frequently employed of research methods, surveys are widely used within software engineering research [Pfleeger and Kitchenham, 2001, Smith et al., 2013] and within SMEs [Dennis, Jr., 2003]. Indeed, survey adoption is considered a well-known strategy when conducting empirical studies [Punter et al., 2003], especially within a software engineering context [Ciolkowski et al., 2003, Cater-Steel et al., 2005]. Aside from helping describe a phenomenon of interest, they help researchers obtain a clearer picture of a product, process or population [Kitchenham and Pfleeger, 2002c]. Most software engineering surveys are cross-sectional, case control studies [Kitchenham and Pfleeger, 2002c], capturing the current status of a situation [Wohlin et al., 2003]. Therefore, they are appropriate for addressing the research question and the associated research objectives of this thesis (this is elaborated upon in Section 4.5).

We now consider our survey approach.

4.3.1 Survey Design Considerations

As it is easy to conduct surveys of questionable quality [Kelley et al., 2003] (including within an empirical software engineering research context [Ciolkowski et al., 2003]), it becomes essential to adopt some form of guiding process. We thus examine [Pfleeger and Kitchenham, 2001, Kasunic, 2005] which focus on conducting software engineering surveys. However, as we perform an online survey, we also refer to [Schonlau et al., 2002, Kelley et al., 2003, van Selm and Jankowski, 2006, Singh et al., 2009] for general guidance and [Punter et al., 2003] for specific advice concerning online software engineering surveys.

Therefore, as suggested in [Bradburn et al., 2004, Kasunic, 2005], we limit scope as too many questions may lead to non-response [Dennis, Jr., 2003, Punter et al., 2003, Porter, 2004] (notably, software developers are more likely to complete shorter surveys [Smith et al., 2013]). We also acknowledge that the type and format of the questions, and the structure of the survey itself, will impact response rates and should be considered to minimise bias [Kelley et al., 2003] and maintain respondent motivation [Punter et al., 2003]. There is a clear relationship between a survey's design and the quality of information which it can obtain [Fowler, Jr., 1995, Daly et al., 1995] i.e. "good research cannot be built on poorly collected data" [Gillham, 2008, p. 1]. Further, and aside from keeping survey design simple [Umbach, 2004], it is important to ensure that the survey is written for the respondent, not the researcher [Kasunic, 2005]. As a practising software engineer, the author of this thesis was well-positioned in creating an instrument which would be understood by like-minded individuals. Nonetheless, an instrument should be pre-tested to determine loading bias, unnecessary complexity [Daly et al., 1995], whether questions are consistently understood and whether any accompanying instructions can be followed [Kelley et al., 2003]. This helps ensure instrument reliability and validity, which is important since software engineering surveys are reportedly weak in this regard [Kitchenham and Pfleeger, 2002a] (see Section 5.1.1 for the pre-testing performed). Further, and similar to [Daly et al., 1995], we allow respondents to submit feedback about the survey if desired (see Appendix H).

Similar consideration is required when identifying the respondents i.e. the sample derived should be representative of the population (so as to avoid bias), able to address the research objectives (i.e. appropriate) and not cost prohibitive [Kitchenham and Pfleeger, 2002c]. Deriving the sample can be approached in either a probabilistic or non-probabilistic manner. Whilst the former affords each member of the population an equal chance of being selected [Punter et al., 2003] (which eliminates subjectivity, resulting in an unbiased sample [Kitchenham and Pfleeger, 2002b]), it demands an understanding of the entire population. In contrast, non-probability sampling is a non-

systematic approach relying on human judgment [Kasunic, 2005] (however, the results of such convenience sampling can be generalised to a subset of the population [Singh et al., 2009]). See Section 5.1.2 for the sampling strategy employed.

Notably, ensuring high response rates to online surveys can be difficult [Kelley et al., 2003, van Selm and Jankowski, 2006]. This has been attributed to the attitudes and characteristics of the respondents e.g. their harbouring privacy or security concerns, to the lack of incentives, computer availability or web literacy, to survey emails being considered spam [Singh et al., 2009]. However, whilst some individuals uncomfortable with technology may not respond we, like [Punter et al., 2003], feel this is not a valid concern when conducting an online software engineering survey. This is supported by the survey studies reviewed in [Schonlau et al., 2002] (which found computer literate respondents were more likely to respond online). Generally, respondents are becoming more accepting of online surveys [Singh et al., 2009]. Similarly, secure websites can address data transmission concerns [Evans and Mathur, 2005] (Qualtrics utilises Transport Layer Security (TLS) encryption [Qualtrics, 2020]). Therefore, whilst survey salience may not be directly under the control of the researcher, a survey's introduction should emphasise its relevance and importance [Porter, 2004] (the importance of demonstrating personal relevance is echoed in [Gillham, 2008]). Certainly, if there is a perceived benefit to society (e.g. that the results will be published), rather than just a private entity, software developers may be more likely to respond [Smith et al., 2013]. Thus, like [Punter et al., 2003], we attempt to motivate respondents through the provision of a report enabling them to compare their state of practice against the population's.

Finally, although surveys can produce considerable amounts of data in a short period of time, such data can lack detail and depth [Kelley et al., 2003]. We thus include open-ended questions and observe that the responses received to such questions are longer for online surveys than mail surveys [Ilieva et al., 2002]. Online surveys also help facilitate the sharing of a respondent's experiences and opinions [van Selm and Jankowski, 2006], with the associated anonymity of a survey resulting in more truthful responses [Daly et al., 1995]. The acquired survey data is also bolstered through a second phase of data collection (see Section 4.4). Notably, conducting follow-up interviews with survey respondents, in order to acquire a more detailed understanding, is an established approach [Ngo et al., 2009]. Therefore, we include a question asking whether respondents would be willing to participate in follow-up interviews (see Appendix H).

Having examined several considerations when employing an online survey, we now justify their adoption.

4.3.2 Justification of Survey Adoption

There are many advantages associated with the adoption of web-based surveys [Singh et al., 2009] - certainly, they have transformed the way surveys are conducted [Evans and Mathur, 2005]. As various data collection methods - including surveys - have been upgraded [Ilieva et al., 2002], embracing this technological change seems not only desirable, but pertinent when considering the intended respondents and subject matter. Therefore, whilst there are advantages and disadvantages with any empirical technique [Singer et al., 2008], surveys are particularly suited for studying software engineering empirically [Daly et al., 1995, Cater-Steel et al., 2005] (with Cater-Steel et al. indicating that they enable the researcher to understand the practices, methods, tools and standards of software engineers - notably, all of which are elements addressed by our own instrument, see Appendix H).

Further support is afforded by the literature reviewed (see Chapter 2), where surveys either completely, or partially, were employed to understand various aspects of V&V within a variety of organisational contexts e.g. [Koomen and Pol, 1999, Koomen, 2002, Cusumano et al., 2003, Torkar and Mankefors, 2003, Stringfellow and York, 2004, Baharom et al., 2005, Runeson, 2006, Wojcicki and Strooper, 2006, Sung and Paynter, 2006, Thörn and Gustafsson, 2008, Rodrigues et al., 2010, Engström and Runeson, 2010, Deak and Stålhane, 2013, Pfahl et al., 2014, Dias-Neto et al., 2017, Pham et al., 2017, Zheng et al., 2017, Santos et al., 2017, Zhang et al., 2018]. Collectively, these examples show that surveys have successfully been used to: evaluate test maturity; understand V&V state-of-practice; and permit focus on both the social and technical aspects of V&V. Further, we observe that respondents have included SMEs (with some surveys focused upon such organisations). Therefore, although concerns have been raised about online survey response rates, we observe they may yield rates comparable, or greater than, those found with paper surveys, however, this is dependent on both the population being surveyed and the design of the survey [Porter, 2004]. As software engineers form our population, the limiting factors of internet access and computer literacy are removed. Further, we have emphasised the importance of good survey design and have highlighted several best practices which we have adopted. Importantly, interest continues to grow in terms of conducting software engineering surveys [Punter et al., 2003], with respondents becoming more accepting of online surveys [Singh et al., 2009].

In summary, online surveys are an effective tool in empirical software engineering, allowing the researcher to understand the state of practice and to identify improvements [Punter et al., 2003]. They can also be used to perform organisational assessments [Singh et al., 2009]. As captured by the thesis research objectives (see Section

1.4.2), we aim to examine security V&V practice within an organisational context, therefore, the utilisation of an online survey appears a sound strategy. Surveys have also been used extensively to understand information security culture within various types of organisation (see Section 4.5.3) and adopted as one of the primary data collection methods when conducting a STIN-based study e.g. [Kim, 2008]. Acknowledging that researchers are generally unaware of what practising software engineers are doing [Cater-Steel et al., 2005], it is only by understanding their practices and perceptions that it is possible to perform research that will have a significant impact [Smith et al., 2013]. Surveys can help provide such an understanding [Cater-Steel et al., 2005, Smith et al., 2013].

We now examine the second, complementary phase of data collection.

4.4 Data Collection Phase Two: Interviews

Interviews aim to develop an understanding of a subject's attitudes, beliefs, opinions, behaviours, processes and experiences [Rowley, 2012] and are a useful way of collecting firsthand information about how software engineering is performed [Seaman, 1999] i.e. they provide insight into a subject's world [Kvale, 1996, Hove and Anda, 2005] and thus an understanding of their perspectives of a specific situation [Perry et al., 2000].

Employing semi-structured interviews, we discount structured interviews as they can result in short answers [Rowley, 2012] and prevent the exploration of interesting responses. Similarly, we discount unstructured interviews since, aside from requiring considerable experience to conduct successfully, the generated interview transcripts can result in data difficult to compare [Rowley, 2012]. This could impact the ability to understand, and identify, emerging themes. In contrast, semi-structured interviews enable a researcher to gather both expected - and unexpected - information [Seaman, 1999] and, by being more interactive, allow researchers to clarify questions, explore responses, and permit a rapport to develop with the interviewee (in turn, improving the quality of the responses received) [Singer et al., 2008]. There is also greater flexibility in question adaptation and question order [Rowley, 2012].

Additionally, semi-structured interviews have been used in many software engineering studies [Hove and Anda, 2005] (e.g. for understanding software test practice [Larusdottir et al., 2010]); employed to understand an organisation's information security culture e.g. [Dojkovski et al., 2007, Ngo et al., 2009, Dojkovski et al., 2010]; and used either as the primary data collection method, or as a subsequent data collection method following an initial survey, in various STIN-based studies e.g. [Meyer, 2007, Kim, 2008, Reinert, 2009, Suri, 2013, Taylor-Smith, 2016, McCoy and Rosen-

baum, 2017, Bingham-Hall, 2017]. Further, having conducted a prior phase of data collection (see Section 4.3), we have a strong basis on which to develop an interview guide. This is important as interviews are both resource-demanding and time-consuming [Hove and Anda, 2005]. Therefore, we now consider our approach when interviewing.

4.4.1 Interview Preparation

Whilst interviews are capable of collecting factual information, their potential as a data collection method "is better exploited when they are applied to the exploration of more complex and subtle phenomena" [Denscombe, 2010, p. 173]. Given the socio-technical nature of software engineering (see Chapter 3), a qualitative approach is necessary to understand the human aspects (such as motivation and communication). However, observing that the quality of the data collected is related to how the interviews are conducted [Hove and Anda, 2005] (which also impacts the quality of the analysis and reporting [Kvale, 1996]), it is important to be aware of the challenges faced whilst interviewing.

Therefore, acknowledging that a study's results are critically linked to the selection of participants [Rowley, 2012], it is important to understand project scope to correctly define the population of interest. We have already defined the organisational and technological contexts (see Sections 1.2 and 1.3), with the first phase of data collection acting as a sampling frame for the interviews. However, one of the challenges in conducting an empirical software engineering study is access to data [Easterbrook et al., 2008]. Thus we do not limit ourselves to just face-to-face interviews (various data collection methods have undergone technological evolution [Ilieva et al., 2002]). For example, semi-structured interviews, conducted over the telephone, are capable of providing rich data for analysis [Cachia and Millward, 2011]. Similarly, email-based interviewing can assist with "the exploration of busy professional participants' reflections on their lives and identities in the midst of their experiences" [James and Busher, 2006, p. 415] ([McCoyd and Kerson, 2006] found that the richness of the data obtained with email-based interviews was comparable to that of face-to-face interviews). Within Section 4.3.1, we observed, in the context of online surveys, that software engineers should be comfortable with such online technology. We believe the same is true in terms of a technologically upgraded interview.

Observing that an interview is a conversation [Kvale, 1996], we avoid questions which can be answered with just a "yes" or "no" (thereby preventing the shortest answer being given [Seaman, 1999]); we also acknowledge the advice available for formulating interview questions more generally e.g. [Kvale, 1996, Rowley, 2012]. Further, as preparation is essential for determining an interview's outcome [Kvale, 1996], we

develop, and pre-test, an interview guide (see Appendix K). Acknowledging the importance of pre-testing, we conduct a number of practice interviews to help refine the interview guide and to prepare as an interviewer [Witschey et al., 2013] i.e. to develop the skills necessary for ensuring good interaction and for helping an interviewee feel comfortable. This, in turn, should result in interviewees more willing to share their experiences (with knowledge evolving through dialogue [Kvale, 1996]) [Hove and Anda, 2005]. See Section 6.1 for the interview pre-testing performed. Further, as semi-structured interviews allow additional questions to be asked as an interview progresses, we, like [Witschey et al., 2013], adopt any such questions if they resulted in useful information. Likewise, questions failing to produce useful information were removed (see Figure 4.1 for where the interview guide may undergo refinement).

As an audio record increases detail richness, ensures accuracy, and permits the interviewer to focus on the interview (as opposed to taking notes), all interviews (including the pre-testing sessions) are recorded. Although some interviewees may feel uncomfortable with this, [Hove and Anda, 2005] indicate that they have never found a software engineer adverse to the use of a "tape recorder". Further, as we employ a variety of mediums when interviewing, we observe that telephone interviews can be digitally recorded [Cachia and Millward, 2011] and that email removes the need for transcription [McCoyd and Kerson, 2006]. Although all data has potential value [Seaman, 1999], data analysis has been described as "time-consuming, difficult, and sometimes unpleasant" [Witschey et al., 2013, p. 53] (with, approximately, one hour of audio data taking three-four hours to summarise and eight hours to transcribe [Hove and Anda, 2005]). Thus we heed the advice of Rowley, namely, that the length and number of interviews conducted needs to be carefully considered to avoid "drowning in a sea of data" [Rowley, 2012, p. 263].

Finally, we acknowledge the importance of demonstrating that the interviewees either have the appropriate authority, or knowledge and experience, to provide useful information [Rowley, 2012]. Similarly, [Hove and Anda, 2005] echo the importance of including information about both the interviews and the interviewees when reporting the results which, they observe, is an aspect often overlooked. The results from the second phase of data collection and analysis are discussed in Chapter 6 and the interviewees are summarised within Appendix L.

Having examined our approach to interviewing, we now justify their adoption.

4.4.2 Justification of Interview Adoption

Much of the phenomena surrounding software development is qualitative in nature (it is a socio-technical discipline, see Chapter 3) and semi-structured interviews are

an appropriate means for collecting such data [Hove and Anda, 2005]. Whilst there are advantages and disadvantages to adopting any empirical technique [Singer et al., 2008], interviews are one of the primary methods for collecting data about a subject's world, their attitudes, beliefs and behaviours (they are often used in empirical software engineering research, with semi-structured interviews being the most common type employed) [Kvale, 1996, Hove and Anda, 2005, Rowley, 2012].

Analysis of the literature reviewed within this chapter (and Chapter 2) clearly shows that interviews comprise the data collection method in a number of V&V studies, and information security culture studies, across a variety of organisational contexts e.g. [Groves et al., 2000, Andersson and Runeson, 2002, Schlienger and Teufel, 2003, Cohen et al., 2004, Kollanus and Koskinen, 2006, Grindal et al., 2006a, Grindal et al., 2006b, Sitnikova et al., 2007, Dojkovski et al., 2007, Ngo et al., 2009, Epstein, 2009, Dojkovski et al., 2010, Larusdottir et al., 2010, Deak and Stålhane, 2013, Riungu-Kalliosaari et al., 2016, Pham et al., 2017, Cruzes et al., 2017, Zheng et al., 2017]. These examples show that interviews have successfully been conducted within SMEs, and employed to understand various aspects of V&V, including: the activities of software testing and software reviews, their maturity, as well as a variety of socio-technical factors. Notably, interviews have been used to understand aspects of security V&V e.g. [Epstein, 2009, Cruzes et al., 2017]. Interviews have also been used to understand an organisation's information security culture e.g. [Dojkovski et al., 2007, Ngo et al., 2009, Dojkovski et al., 2010]. This demonstrates their appropriateness for addressing the research objectives of this thesis. We also observe that other STIN-based studies rely on semi-structured interviews as either the primary means of data collection, or as a subsequent means of data collection e.g. [Meyer, 2007, Kim, 2008, Reinert, 2009, Suri, 2013, Taylor-Smith, 2016, McCoy and Rosenbaum, 2017, Bingham-Hall, 2017].

In summary, interviews are appropriate for understanding the socio-technical realities surrounding V&V. The second phase of data collection builds upon the first phase by acquiring a richer set of qualitative data.

We now examine how the individual research objectives will be addressed.

4.5 Addressing the Research Objectives

Having determined the methods of data collection, we now examine several models, frameworks and existing studies to address the research objectives. Specifically, we present a series of question themes (developed around the key themes of STIN, see Section 3.4) which form the basis of the survey instrument and the interview guide (see Appendices H and K respectively). The question themes, and their relationship

to STIN, are detailed within Appendix D.

4.5.1 Security V&V Practice

As observed within Chapter 1, and highlighted in [Kreeger and Harindranath, 2012], there is a lack of knowledge concerning how security V&V is practiced within SMEs. This has implications, namely, that we cannot simply adopt, wholesale, an existing instrument or approach. However, we can begin by examining several existing studies where aspects of security V&V are touched upon i.e. [Geras et al., 2004, Causevic et al., 2009a, Causevic et al., 2009b, Epstein, 2009, Garousi and Varma, 2010, Larusdottir et al., 2010, Cruzes et al., 2017] (all of which use online surveys (the majority), or conduct interviews, or a combination of both). It is apparent, when reviewing these studies, that there is a noticeable absence of detail. For example, just establishing whether security testing is performed e.g. [Geras et al., 2004, Garousi and Varma, 2010], or only understanding the degree to which the activity is practiced e.g. [Causevic et al., 2009a, Causevic et al., 2009b]. Notably, [Larusdottir et al., 2010] determine who performs the activity and the reasons why it might not be practiced (which, in terms of STIN, at least allows the interactors to be understood and resource flows to be identified). However, aside from security testing, the other security V&V activities are typically overlooked.

Further, organisational size is either not captured e.g. [Geras et al., 2004, Cruzes et al., 2017], or it is, but is not reflected or discussed e.g. [Garousi and Varma, 2010, Larusdottir et al., 2010]. Or the definition adopted is non-standard, for example, [Epstein, 2009] adopts the categorisations of "small", "medium" and "large", but this is based on estimated sales volume and not headcount. Epstein defines a medium organisation as having a sales volume falling between $100 million and $1 billion, in contrast, our adopted SME definition states annual turnover should not exceed €50 million (see Section 1.2.1).

This confirms that we cannot make recourse to an existing instrument when examining security V&V as practiced within UK-based software SMEs. However, several generic software security maturity models exist, which can be used for guidance i.e. the Building Security In Maturity Model (BSIMM) [McGraw et al., 2016] and the Software Assurance Maturity Model (SAMM) [OWASP, 2017] (these are summarised within Appendix M). We make recourse to both BSIMM and SAMM (specifically, the practices associated with the quality control business goal and the verification business function respectively) when formulating a series of questions to address Research Objective 1: "*To examine how security V&V is practiced, supported, and perceived, within UK-based software SMEs*". Notably, both models encompass the activities associated with our

definition of security V&V, namely: security design and code reviews, security testing and penetration testing. It is within Section D.1 that we present the formulated question themes and, where appropriate, show where they have been derived from BSIMM and/or SAMM, as well as their relationship with STIN.

4.5.2 Software V&V Maturity

In contrast to security V&V, several established, and dedicated, maturity models and empirical studies exist in which aspects of software V&V practice are studied and evaluated (however, software V&V activities are only partially addressed by software process improvement models like CMMI [Jacobs and Trienekens, 2002]). Reviewing several of these studies (i.e. [Koomen, 2002, Grindal et al., 2006a, Grindal et al., 2006b, Sitnikova et al., 2007, Park et al., 2008, Kollanus, 2009, Kollanus, 2011, Camargo et al., 2013, Camargo et al., 2015]), we typically find a focus on one aspect of software V&V. For example, a clear divide exists between studies focusing on test-based activities e.g. [Koomen, 2002, Grindal et al., 2006a, Grindal et al., 2006b, Sitnikova et al., 2007, Park et al., 2008, Camargo et al., 2013, Camargo et al., 2015] and those which focus on review-based activities e.g. [Kollanus, 2005, Kollanus, 2011]. In addition, some studies focus on a particular test level, for example, unit testing e.g. [Ellims et al., 2004, Runeson, 2006]. Whilst software V&V maturity has been studied within organisations of varying sizes, the majority of studies focus on both SMEs and large organisations. Although this can permit interesting comparisons, it is not always possible, through the results presented, to draw conclusions regarding a specific category of organisation e.g. [Koomen, 2002, Kollanus, 2009, Kollanus, 2011, Camargo et al., 2013, Camargo et al., 2015].

Notably, surveys and interviews have been utilised to understand the maturity levels of different aspects of software V&V. For example, [Koomen, 2002, Park et al., 2008, Camargo et al., 2013, Camargo et al., 2015] conduct online surveys, whilst [Grindal et al., 2006a, Grindal et al., 2006b, Sitnikova et al., 2007, Kollanus, 2009, Kollanus, 2011] perform interviews. Whilst both methods have successfully provided data for analysis, reviewing the results of these studies often emphasises the importance of adopting a two-phased approach to data collection (as described in Section 4.2). It is also apparent that several of these studies, and others e.g. [Garousi and Varma, 2010], utilise maturity models on which to structure and develop their questions. Therefore, to address Research Objective 3: "*To establish whether organisations can possess a mature security V&V practice without a correspondingly mature software V&V practice*", we utilise aspects of the the Test Process Improvement (TPI) model [Koomen and Pol, 1999], the Business Driven Test Process Improvement (BDTPI) model [de Vries et al.,

2009], Test Maturity Model integration (TMMi) [TMMi Foundation, 2018] and the Inspection Capability Maturity Model (ICMM) [Kollanus, 2011] (these are summarised within Appendix M). Seeking guidance from more than one model is sensible, especially when acknowledging that no common standard, covering all aspects of software testing, exists [Sung and Paynter, 2006]. It is by seeking such guidance that we have a stronger base on which to evaluate an organisation's software V&V maturity (others have followed a similar approach by devising questions around several existing models and standards e.g. [Sitnikova et al., 2007]). Further, and to permit comparison, we employ the same set of questions when examining the practices, and the associated levels of maturity, of both software and security V&V, see Section D.1.

4.5.3 Influence of Information Security Culture

Observing that the purpose of an information security culture is to ensure that information security becomes part of an employees' daily activities [Sánchez et al., 2010], and second nature [Okere et al., 2012], we aim to determine whether an organisation's information security culture influences the security practices employed during software development. This forms Research Objective 2 (see Section 1.4.2). Specifically, determining whether organisations with a strong information security culture have good security V&V practices i.e. that an organisation's information security culture has pervaded their software development lifecycle (and, in corollary, whether organisations with a weak information security culture have poor security V&V practices). Notably, and based on existing organisational culture studies (both generic and information security-focused e.g. [Schlienger and Teufel, 2003, BIAC and ICC, 2004, Sánchez et al., 2010, Connolly and Lang, 2012, Okere et al., 2012]), we hypothesise that if a strong information security culture is present within an organisation (i.e. information security has become a natural aspect to an employees' daily activities), the organisation will have a more mature security V&V practice or, at least, a better practice than organisations with a weak information security culture.

To this end, we need a means to measure and assess information security culture within an organisation, thereby permitting contrast with the results of Research Objective 1 (see Section 4.5.1). Understanding the current state of an organisation's culture is an important step in fostering an information security culture [Okere et al., 2012], therefore, various means for achieving this exist e.g. [Schlienger and Teufel, 2003, Dimopoulos et al., 2004, Kruger and Kearney, 2006, Burns et al., 2006, Dojkovski et al., 2007, Ngo et al., 2009, NIST, 2014, Da Veiga and Eloff, 2010, Department for Business, Innovation & Skills, 2013]. However, whilst [Connolly and Lang, 2012] summarise many of the existing information security culture frameworks, there is no one accepted

approach to its assessment [Schlienger and Teufel, 2003, Okere et al., 2012]. Further, much of the work on developing an information security culture considers large organisations and overlooks the unique aspects of SMEs [Dojkovski et al., 2007, Dojkovski et al., 2010].

Analysis of the literature enables several observations to be made, namely, that many existing instruments are long and complex. We feel that many of the detailed questions would not contribute deeper in terms of deriving a general understanding of whether an information security culture exists within an organisation. For example, [Da Veiga and Eloff, 2010] contains 85 questions which, if adopted wholesale, immediately infringes our noted desire of keeping the survey as concise as possible (see Section 4.3.1). In addition, many of the questions seem more appropriate for an organisation's information security officer and not the "end users" i.e. the software engineers within the organisation. We believe such questions would not result in meaningful answers, a view echoed by [Okere et al., 2012], who indicate that such an approach will only provide the view of the security officer and not that of the wider organisation. Further, whilst some studies examine information security culture within software producing organisations, many studies cover a variety of diverse industry sectors within their analyses and do not present conclusions across individual sectors e.g. [Burns et al., 2006] covers various sectors from tourism to manufacturing. Separating out software producing organisations (as defined within Section 1.2), permits the development of a more focused understanding of information security culture within such organisations. The benefit of this, for example, is that comments indicating that SMEs do not typically have formally documented information security policies e.g. [Dimopoulos et al., 2004, Burns et al., 2006], may be found as being not applicable to software SMEs.

Notably, whilst the majority of existing studies adopt surveys for assessing an organisation's information security culture, some utilise interviews, or a combination of both. The prolific use of a survey further supports its adoption (see Section 4.3.2) - specifically, in terms of understanding an organisation's official and true values [Schlienger and Teufel, 2003], as well as their, and an individual's, current attitudes and values [Ngo et al., 2009]. Therefore, to address Research Objective 2: "*To determine whether organisational information security culture influences security V&V practice*" - we developed our instrument around the guidelines presented within [BIAC and ICC, 2003] (which are relevant to SMEs [BIAC and ICC, 2004]). These guidelines are based upon the Organisation for Economic Co-operation and Development's (OECD) nine principles for developing a security culture [OECD, 2002]. Dojkovski et al. summarise the approaches, detailed within the existing literature, for establishing an information security culture within an organisation: policies, awareness, training and education

[Dojkovski et al., 2007]. We, therefore, ensure these are covered within the presented question themes, see Section D.2.

4.6 Justification of the Methods and Approach

Within this chapter we have detailed the research approach adopted i.e. a sequential explanatory mixed methods design (which is considered a popular design amongst researchers [Ivankova et al., 2006]). Such an approach permits an emergent view to form i.e. the second phase is informed by the first phase. As Venkatesh et al. indicate: "the goal of a sequential research design is to leverage the findings from the first study to inform the second study and add richness to the overall study" [Venkatesh et al., 2013, p. 38]. Further, although it has been acknowledged for some time that quantitative and qualitative research methods can complement one another (e.g. [Jick, 1979]), they have typically been employed in isolation when studying aspects of software engineering and V&V. Whilst both approaches share the same aim (i.e. "to understand phenomena systematically and coherently" [Onwuegbuzie and Leech, 2005, p. 383]), alone neither are sufficient in capturing either the trends, or the details, of a situation [Onwuegbuzie and Leech, 2005, Ivankova et al., 2006, Tashakkori, 2009]. The adoption of a pluralist paradigm approach ensures that a more complete, richer understanding is formed [Mingers, 2001, Johnson and Onwuegbuzie, 2004, Ivankova et al., 2006, Creswell and Plano Clark, 2011, Teddlie and Tashakkori, 2012, Venkatesh et al., 2013, Venkatesh et al., 2016].

Acknowledging that a pluralist approach affords flexibility in research method adoption, with quantitative data permitting the ability to generalise and qualitative data helping explain relationships [Onwuegbuzie and Leech, 2005], we employ a survey within the first phase of data collection and interviews within the second. Both research methods have been used extensively when studying software engineering, and V&V, empirically (although, as noted above, usually in isolation of one another); as well as when investigating information security culture and when conducting STIN-based studies. Chapter 2, and Sections 4.3 and 4.4, attest to their prolific and successful use within these contexts. The adoption of these research methods, when employing a sequential explanatory mixed methods design, is also supported by the mixed methods literature e.g. [Mingers, 2001, Brannen, 2005, Venkatesh et al., 2013].

Whilst it was previously considered very rare for empirical software engineering studies to employ, and report on, the use of multi-methods, it is since acknowledged that utilising a combination of research methods allows for more stable and generalisable results [Mandić et al., 2009]. Notably, there are now many empirical software

engineering studies which have employed a sequential mixed methods design (i.e. utilising surveys in the initial phase and interviews within the second). For example, [Storey et al., 2008] employ such an approach to understand the work practices of software developers and conclude that the interviews, conducted within the second, explanatory phase, allowed a more detailed understanding to develop over the initial, exploratory phase of the study. Similarly, [Chung et al., 2010], when studying the diagramming practices that are utilised in the design of open source software, indicated that a better understanding of actual practice was obtained. Further, when examining the requirements engineering practices, as found within small and medium-sized organisations, [Tahir and Ahmad, 2010] start with a survey and then perform a series of semi-structured interviews to confirm and clarify the answers from the first phase. Whilst the survey allowed them to understand current practice, they indicate that the interviews enabled them to acquire detailed descriptions of software development practices which, in turn, would help with improving such practices (echoing our motivation for studying software engineering empirically, see Section 1.1).

The adopted approach has also been successfully employed within several V&V-related studies. For example, when studying software test practice within industry, [Larusdottir et al., 2010] performed an online survey and then conducted six interviews to strengthen, and exemplify, the results from the survey. Deak et al. used an online survey to acquire an overview of how Norwegian software companies (both SMEs and large organisations) organise their software test activities and then, acknowledging the potential for deeper insights, employed interviews [Deak and Stålhane, 2013]. Similarly, [Pham et al., 2017] performed an online survey to understand how practitioners (also within SMEs and large organisations) perceive inexperienced software engineers when it comes to their software testing skills; they then conducted a number of semi-structured interviews, with the survey participants, in order to acquire a better understanding, as well as deeper insights. Rungi and Matulevičius conducted a TMMi assessment survey, whereupon the results fed into the ensuing interviews [Rungi and Matulevičius, 2013] and, in order to understand the V&V being applied to cyber-physical systems, [Zheng et al., 2017] began with a survey, with the subsequent interviews allowing them to more fully explore, and corroborate, the findings from the survey. They summarise that the study benefitted from each of its parts. Notably, whilst many of the reasons given in these V&V studies echo the reasons for adopting a mixed methods approach in general (see Section 4.2.2), they also help confirm its application to our specific technological and organisational study context, namely, that of V&V within software SMEs.

In summary, mixed methods can be used to "develop a deep understanding of a phenomenon of interest" [Venkatesh et al., 2013, p. 24] (this is also acknowledged within

a software engineering context [Wood et al., 1999]). It is through the combination of quantitative and qualitative research methods that more complete knowledge can be produced, which is necessary in terms of informing both theory and practice [Johnson and Onwuegbuzie, 2004]. A sequential approach - involving the use of multiple methods - should be adopted when researching underexplored areas [Mandić et al., 2009]. As observed within Chapter 1, and [Kreeger and Harindranath, 2012], security V&V, as practiced within UK-based software SMEs, is an empirical unknown. Therefore, mixed methods are appropriate "to holistically explain a phenomenon for which extant research is fragmented, inconclusive, and/or equivocal" [Venkatesh et al., 2016, p. 437]. It is only by adopting a worldview grounded in pragmatism - and employing both quantitative and qualitative research methods - that a more complete, socio-technical analysis can be performed.

Chapter 5

Data Collection and Analysis: Phase One

Within this chapter we present the first phase of data collection. This includes detailing the pre-testing performed, the sampling strategy, the method of instrument distribution and discussion of the response rate. We then analyse the data received and reflect upon the findings and the approach taken. Specifically, this chapter provides:

Section 5.1: We discuss the survey's pre-testing, sampling and distribution strategies.

Section 5.2: The survey response rate is discussed.

Section 5.3: We detail the approach taken when analysing the data obtained.

Section 5.4: Respondent demography is examined to provide organisation and respondent context to the responses received.

Section 5.5: We show how frequently the V&V activities are performed.

Section 5.6: We analyse the state of V&V practice, highlighting the differences which exist between software and security V&V.

Section 5.7: We examine the relationship between an organisation's information security culture and the level of V&V practiced.

Section 5.8: We summarise the findings from the first phase and reflect upon the approach taken.

We begin by detailing the employed survey pre-testing, sampling and distribution strategies.

5.1 Survey Pre-Testing, Sampling and Distribution

Within this section we detail the instrument pre-testing performed and how our sample was determined. We then discuss how the survey was distributed and the respondents contacted.

5.1.1 Instrument Pre-Testing

The importance of instrument pre-testing is well known [Hunt et al., 1982, DeMaio et al., 1998, Presser et al., 2004, Wildy and Clarke, 2009]. Therefore, observing that concurrent and retrospective pre-testing approaches can be combined [Drennan, 2003], with each providing different information [DeMaio et al., 1998], we utilised a combination of both. In summary, we spent just over 15 hours with the respondents (across six concurrent and five retrospective pre-testing sessions): 133 comments were received, 72 were addressed. The comments unaddressed typically fell into one of the following categories: the respondents proposed new options or questions, sought some form of clarification, were duplicates, or were positive and required no further action. In all cases we ensured an understanding was reached with the respondent before discounting any comment (we also revisited the survey with the first two respondents to ensure subsequent changes had not affected the instrument).

The subject matter experts involved were all practising software engineers, based within industry, who had performed V&V activities on average for at least eight years within both SMEs and large organisations. As respondents should "match the characteristics of the proposed sample" [Drennan, 2003, p. 59], this helped ensure that the relevant technical and organisational nuances of the survey were tested by an appropriate set of respondents. Further, three of the respondents had English as a second language, which helped ensure question accessibility.

We summarise the pre-testing performed within Appendix E.

5.1.2 Sampling Strategy

We derived our sample through use of the Financial Analysis Made Easy (FAME) database [Bureau van Dijk, 2020] which, as evidenced by its prolific use in other SME-focused studies, is a well-established means of determining a sample e.g. [Duan et al., 2002, Wong and Aspinwall, 2005, Martínez-Caro and Cegarra-Navarro, 2010, Chang et al., 2011, Sinkovics et al., 2013, Ramdani et al., 2013, Dyerson et al., 2016, Mawson and Brown, 2017]. See Appendix F for the search strategy employed.

As is consistent with others utilising FAME (e.g. [Martínez-Caro and Cegarra-Navarro, 2010, Gök et al., 2015]), we applied some additional refinement. Specifically,

we used an organisation's website to determine product relevance, whether they had recently been acquired or were part of a larger group (we also utilised [LinkedIn, 2020]). Following this manual validation, 160 organisations were identified. To further supplement this probability sample, convenience sampling was employed (many software and V&V studies have utilised convenience sampling e.g. [Wojcicki and Strooper, 2006, Park et al., 2008, Causevic et al., 2009a, Causevic et al., 2009b, Epstein, 2009, Kollanus, 2009, Witschey et al., 2013]). Finally, the survey included a question asking all respondents to indicate their organisation's size (see Appendix H). We did not include a similar question regarding turnover since the type of respondent being targetted was unlikely to have been able to have furnished such information. However, the survey included the following statement: "Please only click the "NEXT" button to begin the survey if your organisation has less than 250 employees and an annual turnover less than or equal to £36 million (€50 million)." Clearly, the sample derived via FAME met the turnover requirement since an organisation's filed account information was utilised.

Having determined the sample, we now detail how the survey was circulated.

5.1.3 Survey Distribution and Contact Approach

Survey distribution occurred in two distinct phases: firstly, we contacted the FAME identified organisations utilising a general company email address. Unfortunately, this approach proved unsuccessful, as no responses were received. The second phase involved identifying and contacting specific individuals at the organisations (both those identified in the probability sample and through convenience sampling). This proved successful in establishing contact and also permitted our initial communication to be tailored (as found, such personalisation positively impacts web survey response rates [Heerwegh et al., 2005, Fan and Yan, 2010]). Email was primarily used to contact the respondents, otherwise, contact was established through LinkedIn (often resulting in a respondent providing an email address for subsequent communication).

Within this initial communication we introduced the survey topic and its value (see Appendix G) and, in terms of personalisation, attempted to demonstrate why respondents had been selected (in several cases, LinkedIn profiles helped identify whether individuals had, for example, an interest in security testing). The responses received broadly fell into one of three categories: a positive response - indicating willingness to participate; a response requesting more information, or seeking additional assurance regarding confidentiality and anonymity; a negative response - with the respondent declining to participate. Several responses fell between the two former categories. We elected not to pursue communication with respondents who actively declined to participate. Therefore, subsequent communication included providing a unique link to

the survey and the stipulation of a survey completion date (see Appendix G). Where necessary, we also tailored the response to address any questions raised (e.g. several respondents sought additional assurance that any published data would be anonymised and that respondent confidentiality would be maintained). The first page of the survey reiterated much of the above, as well as providing some additional detail (see Appendix H).

We now examine our response rate.

5.2 Response Rate Analysis

A survey's topic clearly influences an individual's desire to participate [Tourangeau et al., 2000], with topic sensitivity one of the primary factors resulting in non-response [Kays et al., 2013]. Sensitive questions (e.g. those deemed intrusive, or which could lead to repercussions in the event of information disclosure) impact response rates [Shoemaker et al., 2002, Tourangeau and Yan, 2007]. With this in mind, we acknowledge that information security is one of the most intrusive, and sensitive, types of organisation research [Kotulic and Clark, 2004, Uffen and Breitner, 2015]. Within Kotulic and Clark's paper - appropriately titled: "Why there aren't more information security research studies" - they observe "a general mistrust of any "outsider" attempting to gain data about the actions of the security practitioner community" [Kotulic and Clark, 2004, p. 604]. They attribute this to their failure to achieve an acceptable response rate. Unfortunately, within a UK practitioner context, such difficulties continue to be encountered [Reece and Stahl, 2015].

Non-response can occur at the survey and individual question level [Tourangeau and Yan, 2007]. We believe both influenced our response rate. For example, one potential respondent wanted to "*check for any possible sensitivities*" and requested to see the questions before committing - initially indicating, however: "*in principle we would be glad to take part*". This resulted in additional discussions between the respondent and their manager, and whilst acknowledging that the "*research area sounds interesting and important*" (thus demonstrating topic salience) - and that the survey was anonymous - they concluded: "*for commercial reasons we have to decline to participate*". Similarly, another indicated: "*I don't believe I would be permitted to disclose all methods either anyway*" and thus were unwilling to participate. There were also four individuals who started, but failed to complete the survey - we assume certain questions were deemed sensitive. Certainly, some aspects e.g. those touching upon information security policies and their compliance are considered highly sensitive by organisations [Siponen et al., 2014]. Further, adopting a socio-technical lens introduces the human element, with

STIN explicitly exploring undesired interactions (see Sections 3.3.6 and 3.4). Attempting to acquire such information may have generated information disclosure concerns. However, "the essence of defining sensitive topics is found in the personal appraisal of threat" [Kays et al., 2013, p. 159]. Thus, what one person considers to be sensitive, another may be happy to discuss.

Therefore, we heed the words of Kitchenham and Pfleeger: "[i]f we have a large amount of non-response but we can understand why and can still be sure that our pool of respondents is representative of the larger population, we can proceed with our analysis" [Kitchenham and Pfleeger, 2002c, p. 20]. Thus, like in [Kotulic and Clark, 2004], we believe that our response rate of 18.13% (in terms of the FAME sample) can be explained by our study falling into both an under-researched area, and one involving a sensitive topic. Kotulic and Clark suggest time is better spent targetting organisations, rather than continuing to mass mail instruments, therefore, convenience sampling was employed to further supplement the responses received. This resulted in eight additional responses, and thus a total of 37 responses overall. By way of comparison, other software SME-focused surveys e.g. [Thörn and Gustafsson, 2008, Thörn, 2010] - which obtained 27 and 25 responses respectively (the latter equating to a 13% response rate) - are considered comparable to most other software engineering surveys. Notably, [Singer et al., 2008] found that 5% is the consistent response rate to software engineering surveys.

In summary, a prevalent view among survey researchers is that sensitive topics can significantly impact response rates [Tourangeau et al., 2000]. This is still considered a barrier when conducting empirically-based information security-related studies within an organisational context [Stanton, 2007]. However, notwithstanding our research spanning a sensitive topic, the usual factors of non-response (e.g. time) may have further compounded the situation. Various factors influence web survey response rates e.g. survey length and question wording [Fan and Yan, 2010]. We believe we have mitigated against many of these (see Section 4.3). We also acknowledge that "response rates to technology surveys are notoriously low, averaging somewhere in the teens", as well as that "a 20 percent response rate for a technology survey can be considered high, especially if it involves sampling small firms, and there is potential attrition in the sample through exits or mergers and acquisitions" [National Research Council, 2004, p. 29]. There were multiple instances of such attrition within the identified FAME sample (see Section 5.1.2). Further, restricting organisational scope to SMEs, we acknowledge the impact to expected response rate levels. For example, one UK-focused study, involving SMEs, received a response rate of 16.5% - which was regarded as satisfactory [Antony et al., 2005]. Rasmussen and Thimm acknowledge the difficulties faced when survey-

ing SMEs and consider their response rate of 12.6% as falling within reasonable and expected limits [Rasmussen and Thimm, 2009]. Similarly, [Dyerson et al., 2016] consider their survey response rate of 7.8% as being comparable to other IT-based studies conducted within SMEs. Therefore, we consider our level of response acceptable, a view which is further strengthened when considering the type of respondents reached, namely: experienced, senior employees, situated within software SMEs.

5.3 Approach to Data Analysis

Based on the response rate and the sample size acquired (see Section 5.2), we acknowledge that these values will impact the statistical power obtainable [Dybå, 2003] and thus our approach to data analysis. Therefore, whilst various statistical methods of analysis exist - ranging from the descriptive, to the inferential (e.g. regression and correlation analysis) - we, like others, acknowledge that the response rate obtained influences the type of analysis which should be performed. For example, in a survey to understand the nature of task co-ordination within software development, their sample size of 30 was considered as being "too small for any kind of statistical analysis" [Amrit, 2005, p. 4]. Similarly, within other empirical software engineering studies, it has been felt, given the smaller sample sizes obtained, that it would not be possible to present results with suitable statistical significance e.g. [Hall et al., 2001, Burton et al., 2011, Villela et al., 2014]. Therefore, when examining the impact of organisational size on software process improvement (within both small and large organisations), [Dybå, 2003] required a minimum sample size of 100 in order to ensure adequate statistical power. Notably, and unlike the examples given, our response rate and sample size was greatly influenced by our researching a highly sensitive subject (see Section 5.2).

Therefore, and given the above, we elect to employ a more descriptive approach when presenting our results. Notably, such an approach helps provide an understanding of the data collected [Sandelin and Vierimaa, 2003, Runeson and Höst, 2009] (including the identification of abnormal or invalid data [Sandelin and Vierimaa, 2003]) and permits various assertions to be made about the population itself [Wohlin et al., 2003]. Further, according to [Ciolkowski et al., 2003], a descriptive approach can be used to generalise empirical findings, thereby permitting research-in-the-large. We recall, within Section 4.2.4, our stated intent of adopting an approach capable of supporting both research-in-the-large and research-in-the-past (i.e. an approach that would enable us to examine how security V&V is currently practiced, supported, and perceived, within UK-based software SMEs). Based on this, we find that such an approach to survey data analysis has been adopted by many other empirically-based software

engineering studies (including those focused on V&V e.g. [Runeson, 2006, Wojcicki and Strooper, 2006, Rodrigues et al., 2010, Engström and Runeson, 2010, Deak and Stålhane, 2013]).

However, like others adopting such an approach (e.g. [Hall et al., 2001]), we acknowledge the necessity of conducting additional research to follow-up, and further explore, the findings obtained. Therefore, whilst we employ a descriptive approach within the first phase of data collection and analysis, we do so with the knowledge that we will further substantiate, and build upon the themes identified, by conducting a subsequent phase of data collection and analysis (with the survey thus acting as a sampling frame for the interviews within the second phase [Brannen, 2005]). This echoes one of the primary advantages of adopting a sequential, mixed methods approach to data collection and analysis (i.e. the quantitative data acquired by the initial survey permits a general understanding to develop, with the qualitative data acquired in the subsequent phase further refining and explaining the quantitative data [Ivankova et al., 2006, Creswell and Plano Clark, 2011], see Sections 4.2.2 and 4.2.3). The utility of such an approach has been confirmed, more generally, within other empirical V&V-focused studies e.g. [Larusdottir et al., 2010, Deak and Stålhane, 2013, Rungi and Matulevičius, 2013, Pham et al., 2017, Zheng et al., 2017]. Thus, having based the development of our survey instrument around a number of existing software V&V and software security maturity models, as well as a number of existing, relevant studies (see Section D.1), we make reference to these studies, where appropriate, to support our findings during the presented analysis.

We now examine the results.

5.4 Respondent Analysis

Before addressing the research objectives, we examine respondent demography to provide organisation and respondent context to the responses received. This covers Section 3 of the survey instrument (see Section H.4) and is summarised in Appendix I.

5.4.1 Organisational Context

Approximately three quarters (72%) of the respondents were from medium-sized organisations, the remainder (28%) being small. This is reflected in Figure 5.1. That 14% of the respondents were able to specify the exact number of employees, and no respondent indicated more than 250 employees, confirms that the filtering performed, when determining the sample (see Section 5.1.2), was successful.

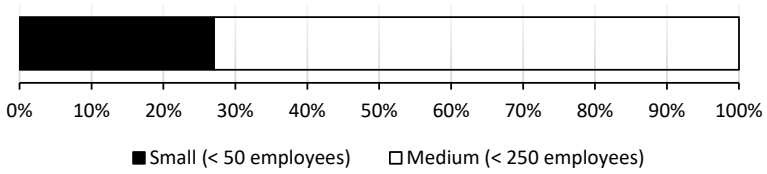


Figure 5.1: Organisational size

These organisations vary in age from 1-2 years to over 20 years, see Figure 5.2. However, utilising the age classes defined within [Criscuolo et al., 2014], the majority are classified as "mature" (24% indicating 6-10 years) or "old" (67% indicating over 10 years). The remainder are either "start-ups" (3% indicating 1-2 years) or "young" (5% indicating 3-5 years). Eight percent of respondents specified the exact age of their organisation (these values were reclassified to the appropriate age class). That 91% of the organisations are either "mature" or "old" adds strength when interpreting the results. Specifically, if the majority were "entrants" (i.e. less than 1 year old) or "start-ups" it would be easy to dismiss any findings as unreliable due to the inherent, chaotic nature of such embryonic organisations.

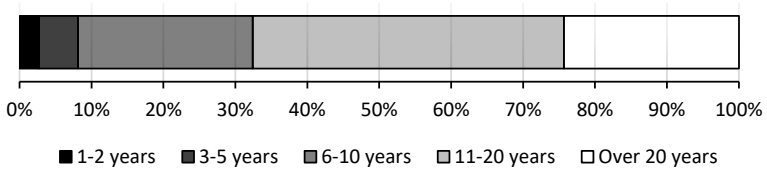


Figure 5.2: Organisational age

Whilst all the organisations develop software products (as defined within Section 1.2), Figure 5.3 shows that they target a variety of industry sectors with their offerings. Unsurprisingly, nearly a third fall under information and communication [Office for National Statistics, 2009], which includes organisations developing products such as secure collaboration systems and secure network management solutions. Importantly, 10 distinct industry sectors are covered, including: financial and insurance activities (e.g. online trading and payment solutions) and the education sector (e.g. parent to school communication). This diversity helps provide further insight and ensures result generalisability. As software security is an ubiquitous concern (see Section 1.1), it would have been unwise to have focused solely on organisations targetting a specific industry sector.

Aside from the differences in organisational size and age - and the nature of their products - we find a variety of software development methodologies being employed (see

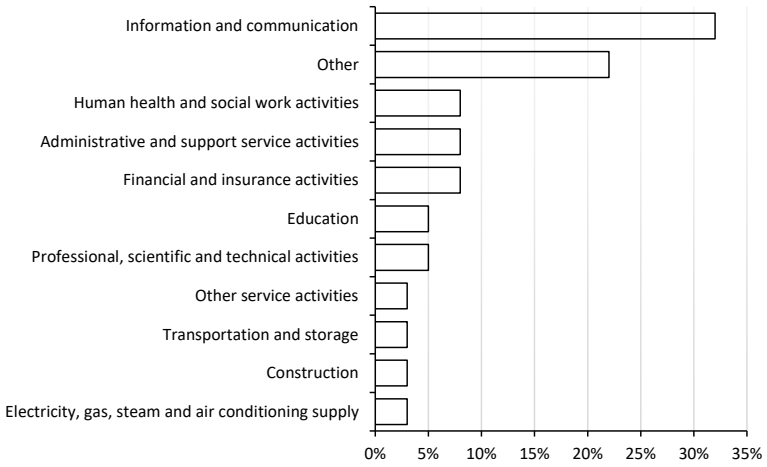


Figure 5.3: Industry sector targetted

Figure 5.4). This influences the nature of the V&V being practiced (see Section 1.3.1). That Agile is practiced in some form by the majority (86%), reflects its popularity [Clarke and O'Connor, 2013, Ryan and O'Connor, 2013, Rindell et al., 2015, Chóliz et al., 2015, Rindell et al., 2017]. Test-Driven Development is the next most widely employed methodology (41%), followed by Waterfall (35%) and then the V-Model (32%). Seventy percent of the organisations employ at least two methodologies (with 21% employing three or four). Where multiple methodologies have been adopted, all but one of these organisations use Agile. Similarly, [Vijayasarathy and Butler, 2016] also observed the use of multiple methodologies within organisations (although regardless of their size), as well as that a strong association exists between the adoption of Agile and organisations with moderate revenues and a small number of employees.

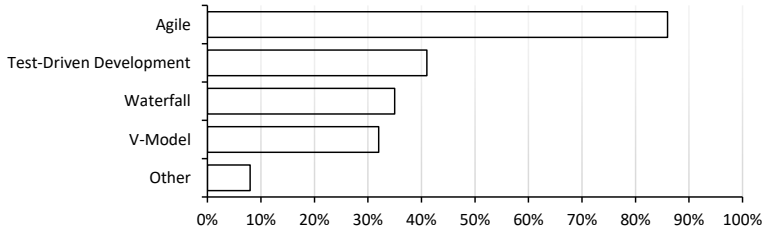


Figure 5.4: Adopted software development methodology

In summary, a good mixture of organisations were surveyed. This diversity is apparent both in terms of their size and potential levels of maturity, as well as their adoption of different software development practices and product focus.

We now examine respondent demography.

5.4.2 Respondent Demography

To ensure an organisation's operational reality is accurately reflected, respondents should be well positioned (although many SMEs have fairly centralised engineering teams, senior employees are preferred due to their wider purview and level of oversight). As indicated in Section 5.1.3, we carefully selected respondents to ensure they were of the appropriate level. Figure 5.5 shows that 81% of the respondents are leads, managers or senior managers, holding positions such as Chief Technology Officer, Head of Engineering, Product Development Director, Test Manager or Senior QA Manager.

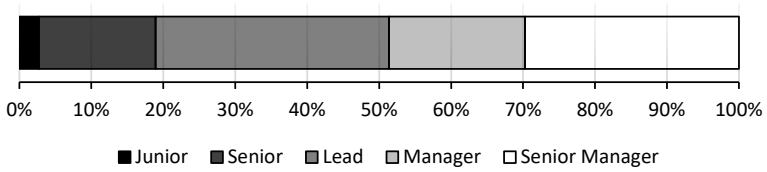


Figure 5.5: Respondent position

That differences exist between developers and test engineers, it was important to ensure respondent diversity to reduce potential bias. Figure 5.6 shows that 62% of the respondents sit within an organisation's test function and 11% within development. Of the 27% indicating "Other", the vast majority sit above test and development, typically assuming a Chief Technology Officer or Head of Engineering role. We consider this a healthy split, ensuring that the test engineer, as a primary interactor, is well represented, and leaving 38% representing development and other technical roles.

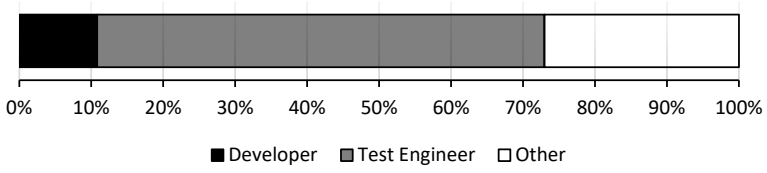


Figure 5.6: Respondent role

Respondent tenure should be considered when interpreting the responses received, since "it helps define an employee's knowledge and perspective on the organization" [Crawford and Leonard, 2012, p. 64]. Figure 5.7 shows that just over a third of the respondents have worked for their current organisation between 3-5 years. The average tenure for a software engineer falls from anywhere under two years [Appelbaum, 2001, TUAC, 2001] to only a few years [Landis, 2011]. Approximately a quarter of the respondents have worked for their organisation for over six years, the remainder less

than three years.

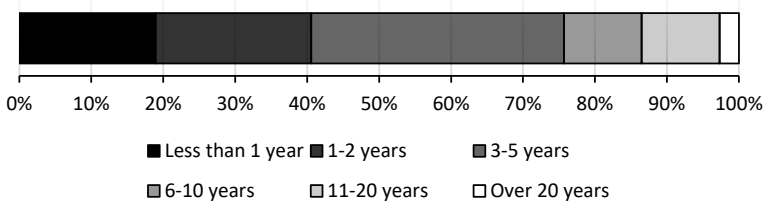


Figure 5.7: Respondent tenure

It is also sensible to consider a respondent's software industry experience as a whole, as well as their software background. This is potentially more important than tenure, as such collective experience allows respondents to develop a more critical eye when examining their current situation. Therefore, as captured within Figure 5.8, it is pleasing that ∼95% of the respondents have at least 6-10 years of experience within the software industry, with ∼62% having over 11 years. Thus, whilst ∼19% of the respondents have worked for their current organisation for under a year (see Figure 5.7), all have at least 3-5 years of experience within the software industry, with the overwhelming majority having at least 6-10 years.

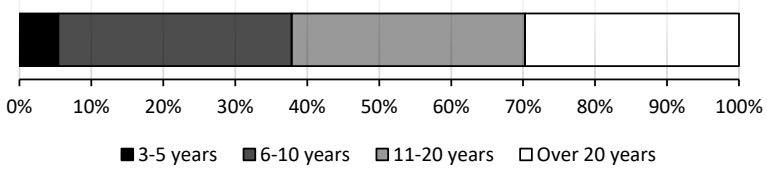


Figure 5.8: Respondent software industry experience

Regarding how this experience has been spent, 70% of the respondents have previously been engaged in test-related roles, 30% in development roles and 22% in other positions (the latter including roles such as support engineer and consultant). This is represented within Figure 5.9. We also observe that only around 8% have held both test and development roles, showing, in corollary, that the majority of respondents have committed to a particular area and have thus built up a respectable level of domain expertise.

Finally, approximately half of the respondents have worked for both SMEs and large organisations, with ∼65% having worked at other SMEs and 81% at large organisations. Only a single respondent had not worked for any other organisation, however, they have worked between 3-5 years for their current organisation. Figure 5.10 shows the types of organisations respondents have previously worked for, adding support that they bring

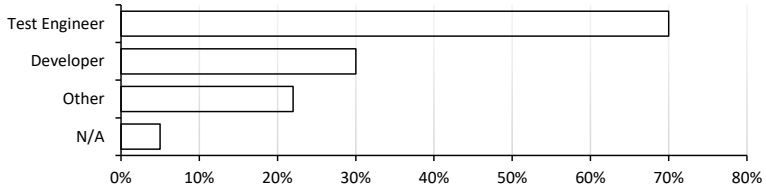


Figure 5.9: Respondent previous roles

both a wealth of experience and a variety of perspectives.

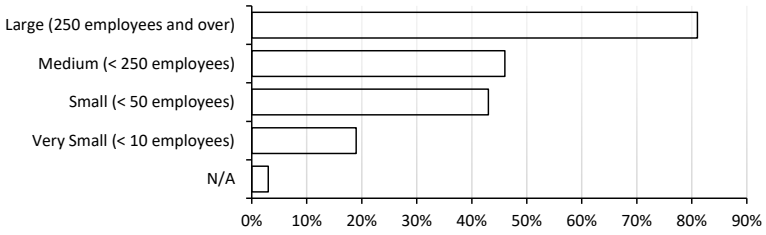


Figure 5.10: Respondent previous types of organisation

Having provided organisation and respondent context, we now proceed to address the research objectives.

5.5 Activity Analysis

We begin by determining whether software and security V&V are performed and how frequently. As Figure 5.11 shows, the software V&V activities are performed with more regularity, within SMEs, than the security-focused activities. If we define "regularly" as an activity which is "Always" or "Often" performed, software testing is regularly performed by all organisations and software review-based activities by ∼68% (only within a few organisations were review-based activities rarely performed, or not at all, or the respondent was unaware whether the activity was being performed). That software testing is the most prevalent activity echoes the findings of others e.g. [Vieira et al., 2006, Ramler and Wolfmaier, 2006, Fernández-Sanz et al., 2009]. However, unlike these examples - which do not reference a specific organisational context - we find this is also the case within software SMEs. That software code reviews are the next most frequently performed activity corroborates [Tian, 2005] but, once again, we confirm this within an SME context.

Notably, contrasting the software V&V activities with their corresponding security V&V activities - in terms of their being performed "regularly" i.e. "Always" and "Of-

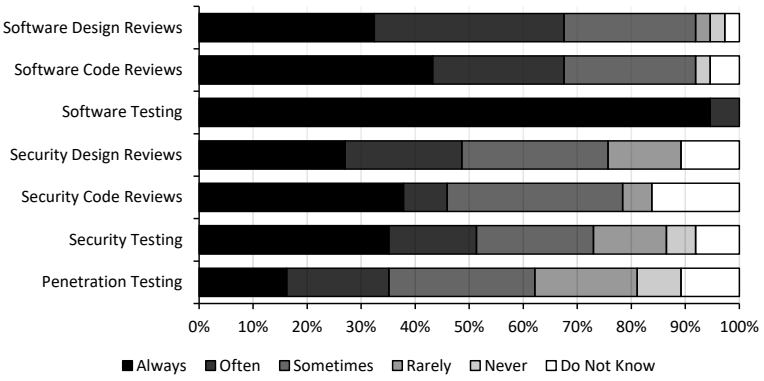


Figure 5.11: V&V activities

ten" - there is a noticeable drop in frequency: review-based activities drop by ∼20% and test-based activities by ∼49% (for security testing) and ∼65% (for penetration testing). There is also a marked increase in the number of "Do Not Know" responses. Whilst such a response can be considered almost equivalent to a "Never" (similar to [Dimopoulos et al., 2004]), we should not discount the possibility that "pockets" may exist within an organisation where the activities are performed. However, as the respondents are predominately senior employees, with respectable tenure (see Section 5.4.2), if they are unaware of whether the activities are performed, we can only infer that such practices are immature, inconsistent, or do not have organisational support. Most capability maturity models would consider this indication of a less mature activity e.g. [CMMI Product Team, 2010].

We now examine V&V practice to address the first and third research objectives.

5.6 V&V Practice

Within this section we examine V&V practice as found within UK-based software SMEs, structuring our analysis through the key themes of STIN (see Section 3.3.6). Specifically, we begin by identifying the human and non-human interactors involved with the V&V activities (Sections 5.6.1, 5.6.2 and 5.6.3), as well as their motivations, and incentives, for performing these activities (Section 5.6.4). In terms of analysing the relationships within the network, we examine the excluded interactors and the undesired interactions which impact the activities (Section 5.6.5). We then examine interactor communication ecology and the resource flows which surround, and influence, the activities (Sections 5.6.6 and 5.6.7 respectively).

5.6.1 Primary Human Interactors

As expected, developers and test engineers are the primary (i.e. more frequently involved) types of interactors engaged across the activities. However, their level of involvement is dependent on activity, see Figure 5.12. Generalising somewhat, developers are much more involved in review-based activities, and test engineers in test-based activities (reflecting developers traditionally being closer to the code and involved earlier in the software development lifecycle).

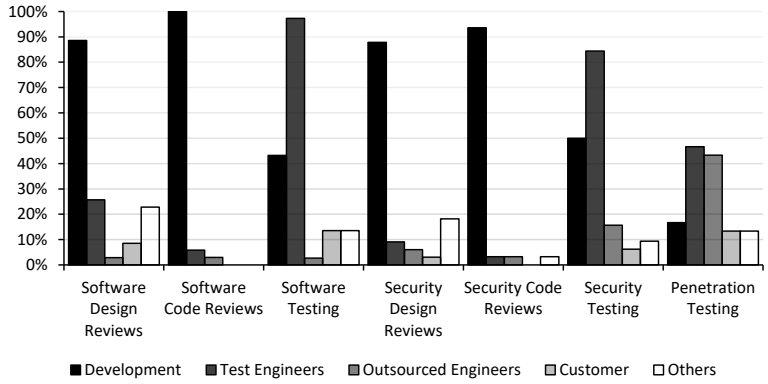


Figure 5.12: Who performs the activities

These pairings (between activity and interactor type) are mirrored when examining activity ownership. As Figure 5.13 shows, developers are more likely to own review-based activities, and test engineers, test-based activities.

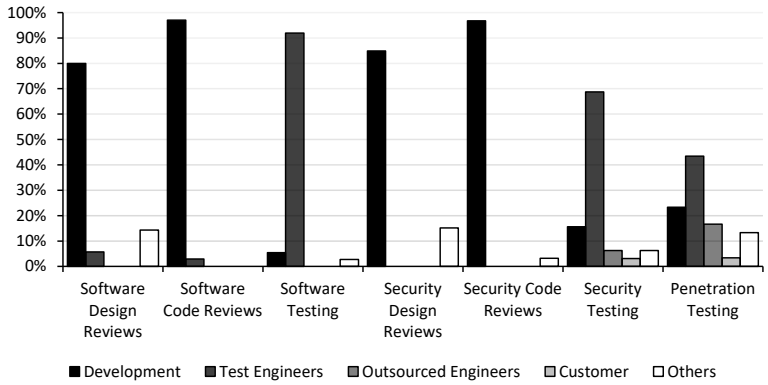


Figure 5.13: Who owns the activities

Interestingly, all organisations distinguish between developers and test engineers. This was somewhat surprising given the wide adoption of Agile (which advocates the generalist [Meszaros and Aston, 2007]), however, many of the organisations practising

Agile also adopt other methodologies (see Section 5.4.1). This conforms with the author's experience within an organisation recently embracing Agile - namely, that the distinct roles of developer and test engineer, whilst blurred in some cases, are still actively acknowledged. Whilst this independence is suggestive of a more mature organisation (as considered by various maturity models e.g. [TMMi Foundation, 2018]), it can potentially result in undesired interactions (see Chapter 3). However, developers and test engineers can, and do, perform the same activities within an organisation. For example, a QA Director [O:WS-I:WE] indicated: "*[d]esign reviews involve technical experts from the Dev team and Product Experts from the Test team to ensure designs are understood, practical, possible, and there are no collisions with other aspects of the products*" (it is telling that this was stated just in terms of software design reviews and not security design reviews).

However, contrasting the actual situation within an organisation, as portrayed above, with the desired situation, we find some interesting results. For example, whilst we observed some involvement of test engineers in what are traditionally developer-led activities (e.g. software code and design reviews), there is an increase in view that they should become involved with these activities (see Figure 5.14). Such involvement, and the resulting improved product knowledge, is critical, otherwise a black-box approach will be adopted by the test engineer - effective security testing requires more than this [Thompson et al., 2002, Potter and McGraw, 2004]. Further, and based on the author's experience, such knowledge can help reduce conflict around defect reporting (an area of contention between developers and test engineers [Cohen et al., 2004]). In corollary, there is no expressed desire for developers to become more involved with software testing, which reflects the findings of others e.g. [Runeson, 2006] and supports the observed detachment between development and testing [Bertolino, 2003].

Similarly, when adopting a security-focused context, test engineers should become more involved with review-based activities. However, and in contrast with software testing, developers should become more engaged with security-focused test-based activities, which perhaps acknowledges that to perform these activities successfully a detailed understanding of the product is required.

Regarding software V&V activity ownership (see Figure 5.15), there is very little difference between the current situation and the desired i.e. developers own, and should own, review-based activities, test engineers own, and should own, test-based activities. However, for the security-focused activities there is an increased view that test engineers should own the test-based activities (and, in several organisations, actually take ownership of the review-based activities), with a decreased view that developers should own these activities. The latter should not be viewed negatively, since Figure 5.14 shows

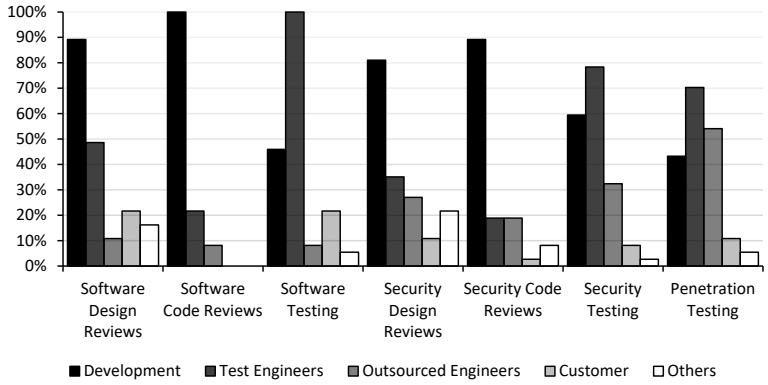


Figure 5.14: Who should perform the activities

that developers should become more involved in performing these activities (thus it is not an indication that only test engineers should perform test activities within an organisation). However, more interesting is the increased view that security-focused activities should be owned by an outsourced agency.

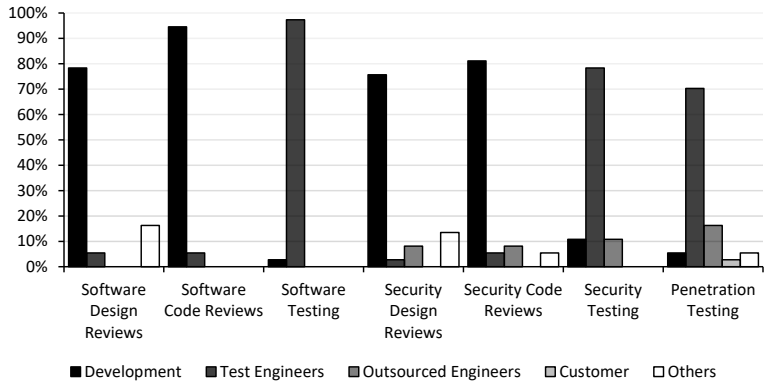


Figure 5.15: Who should own the activities

We now consider the secondary human interactors.

5.6.2 Secondary Human Interactors

Several, lesser involved interactors exist within the V&V socio-technical network. As Figures 5.12 and 5.13 show, there is a greater dependency on outsourced engineers for the security V&V activities than the corresponding software V&V activities (for the advantages and disadvantages of V&V outsourcing, see [Karhu et al., 2007, Smuts et al., 2010]). Significantly, there is a desire to further increase their use - both in performing (see Figures 5.12 and 5.14) and owning (see Figures 5.13 and 5.15) - the security-focused

V&V activities. There is also an increase in outsourcing the software V&V activities, but not regarding their ownership. Thus, whilst there is a belief that more V&V activities should be outsourced, the increase is more significant in terms of security V&V. This is a clear indication that SMEs have greater confidence in performing, and owning, their software V&V activities.

Customers are involved in design reviews and test-based activities although, as one Chief Technology Officer [O:CS-I:GB] observes, this depends on the customer: "*[p]en testing is carried out internally, but for some defence customers and partners, they will also carry out their own independent pen tests and feed their findings back*". That customers are absent from code review activities is not surprising. In addition, we only found two instances of customers owning activities, both concerning security V&V, e.g. a Chief Technology Officer [O:DS-I:SW] indicated this with regards penetration testing (and highlighted this was the desired situation). More generally, there is a desire to increase customer involvement when performing the activities. With the wide adoption of Agile, this is unsurprising given the importance it places on customer involvement [Myers et al., 2011]. However, although Agile is widely adopted, only two organisations (both medium-sized), made reference to product owners: with a Development Manager [O:EB-I:MH] indicating their product owners already assist other interactors with software and security-focused V&V activities; whereas, a Quality Assurance Team Lead [O:CH-I:ZG] states they should help with software design reviews.

Notably, a Product Development Director [O:SP-I:WS] indicated that their internal infrastructure team "*would be involved to advise on the type of security issues we might hit*" when it came to performing security design reviews. This is a good example of not relying on their core, engineering team, but utilising the skills available within the wider organisation (which is clearly important when acknowledging the limited resources of SMEs, and may be suggestive of employees wearing multiple hats [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018]). However, there is still room for further engagement of the infrastructure team as the respondent indicated they should assist both developers and test engineers with the penetration testing currently being performed - a view which seems eminently sensible. Significantly, only a very small minority of organisations actively involve employees outside of engineering, but who have security-focused roles, with their security V&V activities. For example, a Head of Applications Development [O:OS-I:ST] indicated that they have certified employees performing penetration testing, emphasising twice that these employees, whilst internal to the organisation, are independent to development. One Test Manager [O:RW-I:RL] stated that it is their Chief Security Officer "*who owns and manages that the necessary*

verification and validation occurs (whether it be at a customer site or in our offices)". This respondent also indicated, in terms of security design reviews: "*I think that the Chief Security Officer should be involved*", thereby highlighting, as the infrastructure team example above, that more could be achieved in how employees are utilised when it comes to security V&V. Whilst these examples show that several organisations have acknowledged, and utilised, the wider security expertise available within their organisation (although perhaps not as effectively as possible), the lack of respondents indicating even this level of utilisation, or similar, shows improvements are required in how security V&V is practiced, supported, and perceived, within UK-based software SMEs.

Finally, several non-technical interactors were found within the V&V socio-technical network (like [Shachaf and Rosenbaum, 2009], we distinguish between technical and non-technical interactors). For example, within one organisation [O:CS] product management assists with software design reviews but, notably, not with security design reviews (yet another example highlighting the different skill and mindset required). We also found, in several cases, that an organisation's support and sales teams would perform some degree of software testing. Although potentially indicative of employees wearing multiple hats, the fact that the majority of respondents indicating this work for medium-sized organisations, we are inclined to believe this is simply "good practice" by involving those who interact directly with customers (in turn, improving their knowledge of the product and helping feed back customer comments). This view is supported by one Head of Quality Assurance [O:AV-I:JG], who explicitly indicated: "*[t]echnical pre/post sales also perform an element of beta testing*", as well as by a Test Manager [O:RW-I:RL]: "*everyone is responsible for product quality*".

5.6.3 Non-Human Interactors

As STIN acknowledges non-human interactors (see Section 3.3.6), we, like [van der Merwe, 2010, Taylor-Smith, 2016], include tools and processes within our analysis.

5.6.3.1 Supporting Tools

Acknowledging the importance of tools and automation technologies (see Section 2.4), we begin by identifying whether tools are utilised. Figure 5.16 shows that test-based activities are generally better supported by tools than review-based activities and that the software V&V activities receive better tool support than the corresponding security-focused activities.

These findings are also reflected when discussing the tools used. Specifically, respondents were more readily able to identify the tools utilised for software V&V than

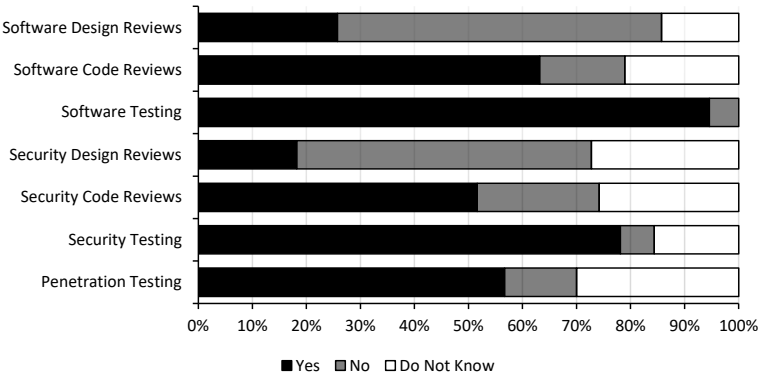


Figure 5.16: Tool presence

their security-focused equivalents. Tools were also better enumerated for test-based activities than review-based activities. Further, whilst a considerable number of tools were referenced, several were in use within multiple organisations. For example, for software testing, there was significant use of the open source tool *Selenium* (a collection of tools for testing web-based applications). There was also frequent use of the load and performance test tool *JMeter* (also open source), as well as a variety of open source xUnit test frameworks and programming languages (e.g. *JUnit*, *NUnit* and *Python*). Notably, some of these are also employed when performing security testing and penetration testing. Regarding these security-focused test-based activities, there was frequent mention of web-related security tools (such as the open source *OWASP ZAP*, and the proprietary offerings *Burp Suite* and *Acunetix*). There was also use of the well-known proprietary vulnerability scanners *Nessus* and *metasploit*.

Generally, there was more use of open source tools than proprietary ones (see Figure 5.17). However, several organisations use Hewlett-Packard's test automation offerings e.g. *Unified Functional Testing* and *Quality Center*; as well as Microsoft's *Test Manager* and *Team Foundation Server*. Further, several organisations employ Atlassian's products e.g. *Jira* and *Crucible* and Synopsys' code analysis tool *Coverity*. Figure 5.17 also shows a fairly healthy amount of in-house tool development i.e. ∼54% of the organisations developed tools in-house to satisfy at least one V&V activity. In contrast, [Sung and Paynter, 2006] indicate ∼30% of the organisations they surveyed developed test tools in-house (as their sample also included large organisations this probably accounts for the smaller amount of in-house tool development). Interestingly, for each activity, the sum of respondents using a combination of open source tools, and those developed in-house, is greater than the reported use of proprietary tools. This echoes the findings of others regarding tool use within SMEs e.g. [Andersson and Runeson,

2002, Sitnikova et al., 2007].

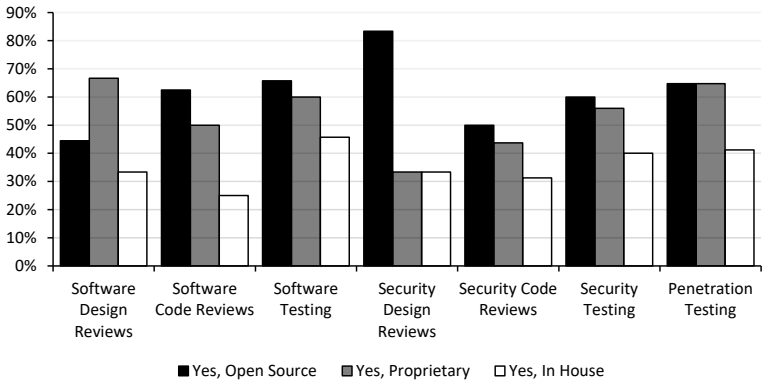


Figure 5.17: Tool origin

Respondents were also less likely to comprehend, or able to describe, the situation when discussing the security-focused activities i.e. the number of "Do Not Knows" increases on comparison with the corresponding software V&V activities (see Figure 5.16). This indicates (and is supported by capability maturity models e.g. [CMMI Product Team, 2010]) that the security-focused V&V activities sit at a lower level of maturity. Notably, a Development Manager [O:EB-I:MH] indicated that their penetration testing is "*performed by external consultants*" and thus they are "*not sure which tools are used*". Similarly, a Test Manager [O:MC-I:AP], regarding both security testing and penetration testing, noted: "*[w]e outsource this to specialists who use a combination of tools. I don't know the names*". With the increased desire to outsource V&V activities (see Section 5.6.2), and security-focused activities in particular, this situation can only worsen.

5.6.3.2 Governing Processes

Having observed the importance of governing processes (see Section 2.5), we find that the software V&V activities are more likely to have processes in place, see Figure 5.18. There is also a noticeable increase in the number of "Do Not Know" responses when contrasting the security-focused activities with their software V&V counterparts.

This emphasises that security V&V, as practiced within software SMEs, sits at a lower level of maturity when compared with software V&V. This view is supported when observing that "[p]ractice without process tends to become unmanageable" [Brown and Duguid, 2001, p. 94]; with various maturity models (e.g. [CMMI Product Team, 2010, Kollanus, 2011]) explicitly indicating that a defined process indicates a higher level of maturity.

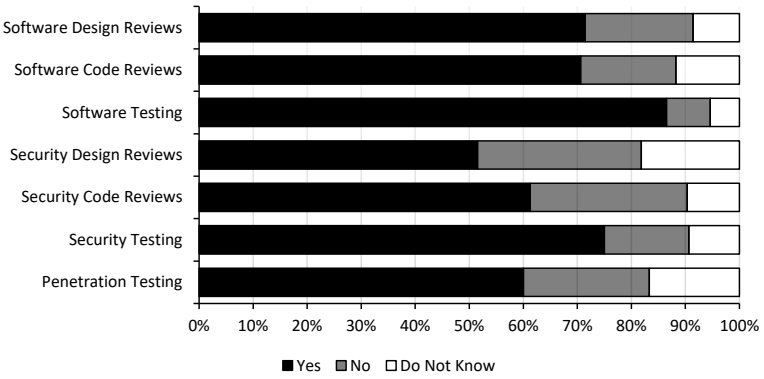


Figure 5.18: Process presence

Having identified the interactors, we now identify their incentives.

5.6.4 Incentives

The next step when employing STIN involves understanding motivation. The importance of this, in a software engineering context, is discussed within Section 3.2. However, as both a complex and poorly understood topic [Sharp et al., 2009], we venture that it is even less well understood in a V&V context. Therefore, we now examine what motivates the interactors.

5.6.4.1 Improving Software Quality

Given the clear relationship between motivation and software quality [Sharp et al., 2009, Sharp and Hall, 2009] we find, overwhelmingly, that people are motivated to perform software and security V&V through a desire to improve software quality (see Figure 5.19). Whilst an often overlooked motivator is the engineering work itself [Tanner, 2003], task identification is the most frequently reported motivator for software engineers [Beecham et al., 2008], which includes "producing an identifiable piece of quality work" [Hall et al., 2008, p. 93]. However, when contrasting the corresponding software and security V&V activities, we find a noticeable decrease in such motivation being reported. Specifically, security review-based activities drop, on average, by 14%, security test-based activities by 35%. The most noticeable drop is when contrasting software testing with penetration testing, which potentially reflects the "too little, too late" view [Arkin et al., 2005, p. 84] i.e. a failure to view security as an emergent property which should be considered throughout software development. Further, this observed decrease, when it comes to security V&V in general, may also be a symptom of respondents viewing security as "just another quality attribute" and failing to

afford importance to it - as a Head of Testing [O:WK-I:RR] indicated: "*I would say we generally have the (in my opinion, wrong) opinion that if customers want to test our products for security issues they can, and we generally only perform security testing when forced*". Whilst we agree that the priority and importance of each quality attribute is dependent on context [Microsoft Patterns & Practices Team, 2009], the significant costs which can result when failing to consider security (see Section 1.3.4) should not be overlooked.

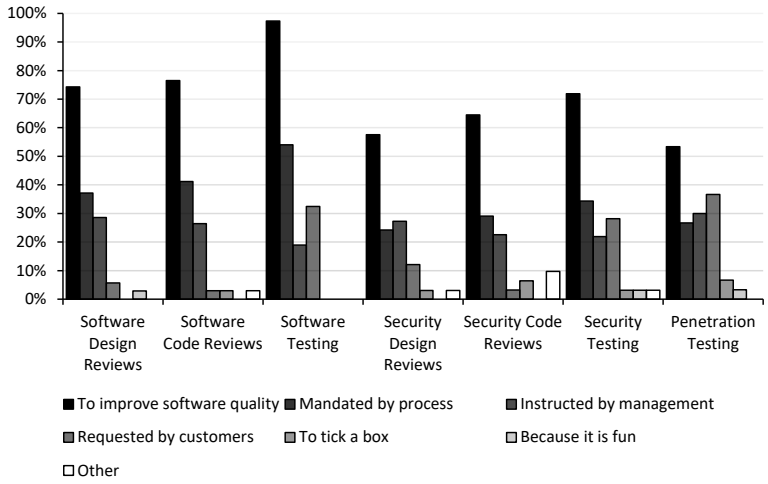


Figure 5.19: Motivation within the socio-technical network

5.6.4.2 Mandated by Process and Instructed by Management

The presence of a governing process is the next most prevalent motivator, although more so for the software V&V activities (on average, 13% more likely for review-based activities, 24% for test-based activities). This is probably a partial reflection that these activities are more likely to have a governing process (see Section 5.6.3.2), which acts as a further reminder that the security V&V activities are less mature.

The results also show that the presence of a governing process is, by itself, not a strong motivator for performing V&V. Specifically, where processes exist - and cover a specific activity (see Section 5.6.3.2) - Figure 5.20 shows, on average, only ∼51% of the respondents are motivated to perform such activities based on the existence - and, more importantly, following their acknowledgement of its existence - a governing process. Further, where processes exist, they are more likely to motivate in terms of software V&V (averaging 58%), than security V&V (where the average drops to 46%). These levels of process acknowledgement - and adherence - provide further indication that the software V&V activities have reached a higher level of maturity.

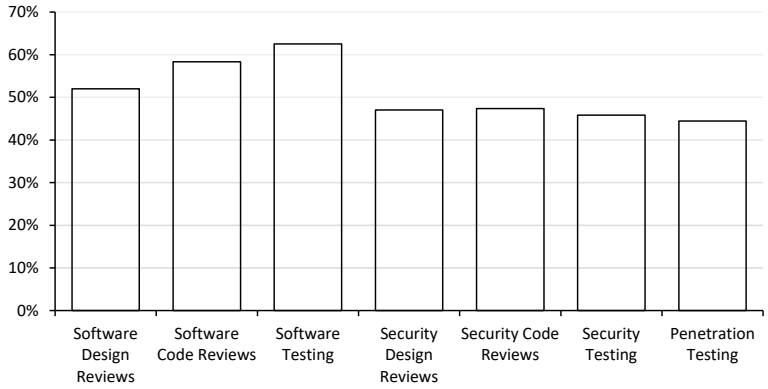


Figure 5.20: Motivation by process

The existence of a process acknowledges that an activity has importance and worth, it also demonstrates management level support for the activity (since a process typically requires management approval before being introduced [Watkins and Mills, 2011]). However, whilst software engineers are required to follow processes [O'Regan, 2011], they usually require enforcement [Testa, 2009]. Notably, management involvement and commitment to V&V is important e.g. trustworthy code reviews can only begin once management has created an appropriate environment [Nelson and Schumann, 2004], and a lack of management support can result in test engineers losing both motivation and interest [Perry, 2006]. Therefore, if management is requesting an activity to be performed, it indicates the activity has their support. However, knowledge workers in the software industry do not respond well to top-level dictates on how to work [Conradi and Fuggetta, 2002]. This supports the observed low response rate of being motivated by management, see Figure 5.21.

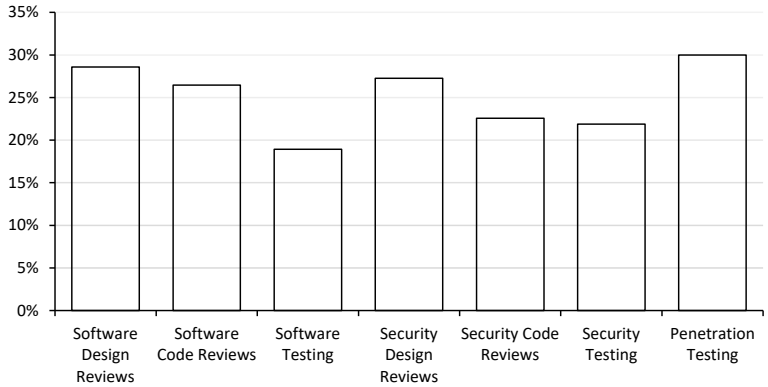


Figure 5.21: Motivation by management

5.6.4.3 Because it is Fun

Tasks considered less enjoyable, tedious or mundane can be enjoyed more readily by introducing the intrinsic motivation of fun [Teh et al., 2013]. However, some V&V activities have received such negative labelling e.g. with developers considering testing as boring and the least enjoyable aspect of their work [Rooksby et al., 2009]. With this in mind, only two respondents indicated the various V&V activities are inherently fun to perform: a Senior Test Development Engineer [O:SC-I:CB] (who finds software design reviews fun) and a Lead Integration Engineer [O:AG-I:DS] (who considers security testing and penetration testing fun activities). The scarcity of such responses is surprising since, although test-based activities have been considered fun [Patton, 2005, Hass, 2014], they are not perceived as such by those practising within SMEs. Interestingly, these respondents are neither recent graduates, nor have they recently started at a new organisation. Further, both their organisations have processes governing the activities, therefore, we also infer that the activities have not recently been introduced and that they have some level of management support and commitment. Therefore, whilst instilling an element of fun into V&V is being explored (e.g. [Dietl et al., 2012]), our results suggest more research is required. They also help support the findings from several other empirical studies showing that people perceive certain V&V activities negatively e.g. [Rooksby et al., 2009, Shah and Harrold, 2010].

5.6.4.4 Ticking a Box

Very few respondents are motivated solely to "tick a box", see Figure 5.22 (as a sole form of motivation it is unlikely to result in V&V being approached with an attitude actively engendering software quality). Specifically, only three respondents are motivated to perform a subset of their activities to simply "tick a box": a Senior QA Manager [O:CR-I:MC] indicated this is their only reason for performing software and security code reviews; a Senior Test Lead [O:PB-I:SK] indicated this is their sole incentive for performing security-focused review and test-based activities; and a Development Manager [O:BN-I:AC] is motivated to perform penetration testing to "tick a box" - as well as by a desire to improve software quality, to adhere to a governing process and to satisfy customer requests. Notably, all of these respondents are motivated by a variety of other reasons for performing the other V&V activities practiced within their organisations.

In summary, whilst the number of respondents indicating they are motivated to perform an activity simply to "tick a box" is pleasingly low, the results show they are more likely to be motivated in such a manner, and typically solely by such a factor,

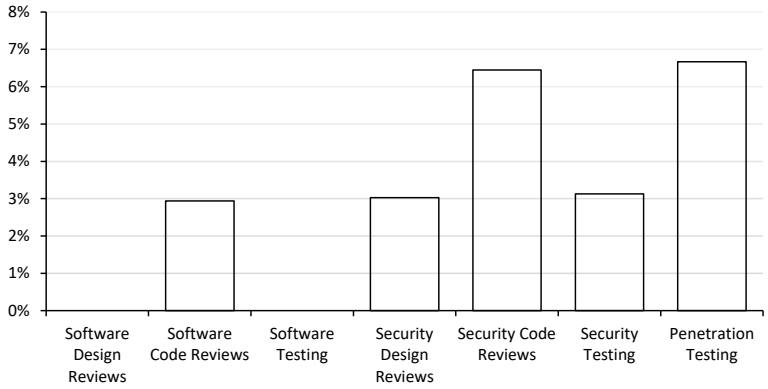


Figure 5.22: Motivation to "tick a box"

when it comes to the security V&V activities. In the case of penetration testing, there is evidence to support this finding since many use it for tick box compliance only [Blackwell, 2014].

5.6.4.5 Customer Request

Figure 5.23 shows customers are a stronger motivating force in terms of test-based activities; thus supporting that some aspects involve customers (e.g. site acceptance tests [Dustin et al., 1999]), with some customers requesting information on an organisation's test maturity and test process improvement activities [Hass, 2014].

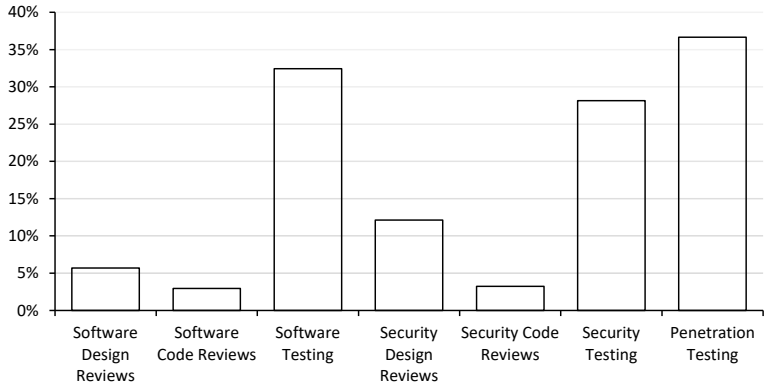


Figure 5.23: Motivation by customer request

The widespread adoption of Agile (see Figure 5.4), an approach advocating customer interaction and involvement (see [Boehm and Turner, 2005]), has also probably contributed towards this. Interestingly, similar levels of motivation are found for the software and security-focused test-based activities. That we typically see a distinction,

when comparing the activities, warrants further explanation. Specifically, for some types of product, deployed within specific contexts, there is an obligation to undergo security evaluation e.g. with governments and financial institutions being required to use FIPS 140 certified products [Smith, 2015] and, within a health care setting, Common Criteria certified products [Mercuri, 2004]. As seen within Section 5.4.1, such industry sectors are well represented within our sample. Additionally, several organisations, who target the information and communication sector, have government customers (this was captured by one organisation's Chief Technology Officer [O:CS-I:GB]) or they produce security software (which is increasingly undergoing security evaluation [Smith, 2007]).

Having identified the interactors and their incentives, we now analyse the relationships within the network.

5.6.5 Excluded Actors and Undesired Interactions

We begin network relationship analysis by identifying the actors excluded from specific activities. To achieve this we observe the differences between the current situation within an organisation (i.e. through identifying the interactors currently performing an activity) and the desired situation. This comparison highlights (see Figure 5.24), that there are more instances of interactors, regardless of type, being excluded from the security-focused V&V activities than the corresponding software-focused activities. Further emphasising the split between interactor type and activity, we find test engineers are more likely to be excluded from review-based activities than developers. Conversely, developers are more likely to be excluded from test-based activities than test engineers. Aside from reflecting the traditional working practices of these interactors, and their typical role division, it also reflects a desire to move away from this long-established separation. The significant adoption of Agile (see Figure 5.4) may have helped encourage this thinking i.e. by blurring the roles [Crispin and Gregory, 2009].

However, overall, there are more instances of test engineers being excluded from activities than developers which, once again, emphasises the power imbalances between the two primary interactors. Whilst the test engineer was the most excluded type of interactor, outsourced engineers were the second most excluded, followed by developers. Figure 5.24 also reflects the desire to increase the reliance on outsourcing (see Section 5.6.2) which, significantly, is more pronounced in terms of the security-focused activities.

It is also possible to identify whether a desire exists to exclude specific types of interactors from performing an activity (this is achieved by identifying the interac-

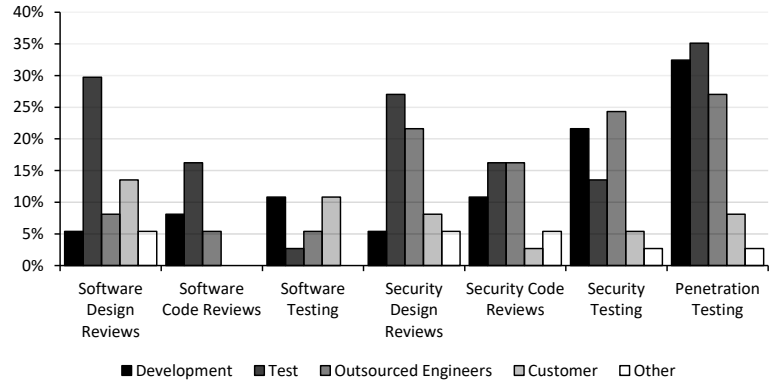


Figure 5.24: Excluded actors

tors currently performing an activity, but which are not subsequently referenced when the respondent describes the desired situation). Figure 5.25 shows, once again, more instances of exclusions - in this case desired, as opposed to undesired - around the security-focused V&V activities.

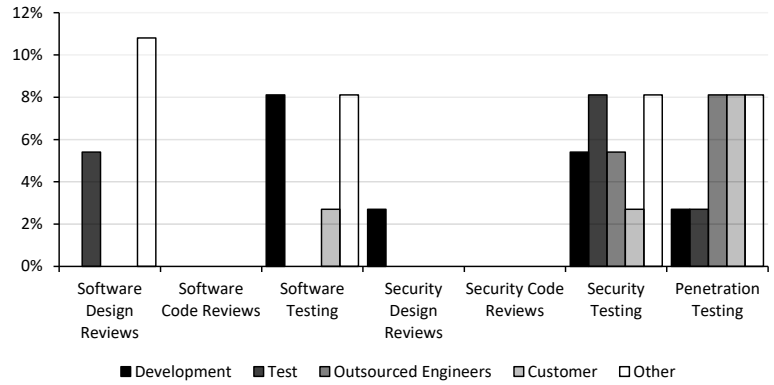


Figure 5.25: Who should be excluded

Interestingly, and despite the noted trend of increasing reliance on outsourced engineers (security-focused activities in particular, see Section 5.6.2), we find that whilst there are no instances of excluding such engineers from the software V&V activities, there are several instances in terms of the test-based security-focused activities. We view this as a desire to remove such reliance and to bring the activities in-house where, in the majority of cases, the respondents indicate that responsibility should fall onto both the primary interactors. This is a promising sign (albeit, far from significant) that, within these SMEs at least, there is intent to improve their in-house security V&V capability level.

5.6.5.1 Resourcing Constraints

Within the vast majority of the organisations (some 89.19%), at least one activity suffers due to a resourcing constraint (within the remainder, 10.81%, no resourcing constraints are found across any of the activities practiced - notably, all of these are medium-sized organisations, classified as "mature" or "old" in age, which may explain why they are less impacted by resourcing constraints). That all activities practiced within 54.05% of the organisations are impacted by some form of resourcing constraint clearly indicates that undesired interactions exist, and surround, the V&V activities, see Figure 5.26.

Overall, a lack of time is the most frequently reported resourcing constraint, impacting software and security V&V. This echoes the finding in [Park et al., 2008] where, in a survey of similar sample size (although different organisational context), a lack of time was the most prevalent issue impacting testing (accounting for 46.97%). In contrast, 67.57% of our respondents indicated that a lack of time impacts their software testing. Notably, Park et al. do not distinguish between the different types of testing, therefore, if we include security testing and penetration testing as well, our average changes to 52.46%. Similarly, within [Larusdottir et al., 2010], a lack of time is the primary factor impacting both software and security-focused test-based activities; and, within [Rodrigues et al., 2010], time is the primary factor negatively influencing the institutionalisation of a test process. However, where our results differ to those above, is by showing that a lack of time is also the principal factor within software SMEs, negatively impacting not only software V&V, but security V&V as well.

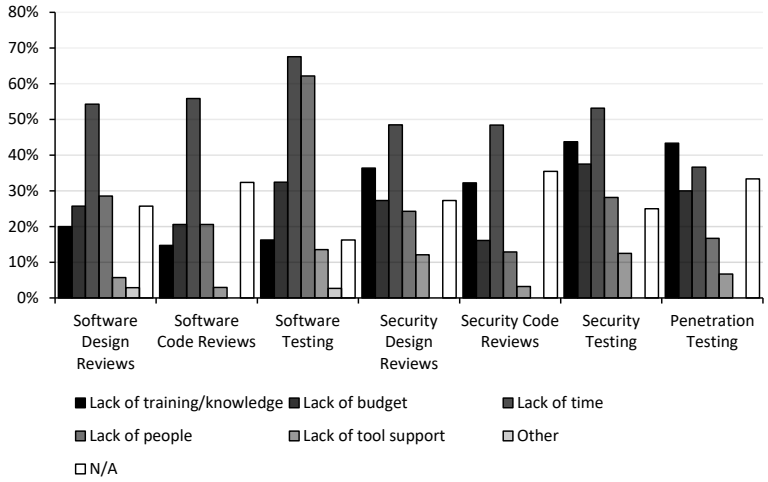


Figure 5.26: Resourcing constraints

The next most frequently reported resourcing constraint shows divergence in how

the software and security-focused activities are impacted. Specifically, a lack of people impacts software V&V whereas, a lack of training and knowledge impacts security V&V. This is a notable finding, and whilst we recall that discovering vulnerabilities is complex [Austin and Williams, 2011], it highlights a need for improved training within SMEs (see Section 5.6.7). Although [Larusdottir et al., 2010] only consider security testing, 47% of their respondents indicate that a lack of training and knowledge impacts security testing - this can be contrasted with our reported 43.75% for security testing. Both we and Larusdottir et al. find that of all the activities studied, security testing is the activity most impacted by a lack of training and knowledge. Whilst their sample is smaller than ours, and we can only state with certainty that 61% of their surveyed organisations are SMEs, the results do help support one another as both are performed within software organisations. Additionally, [Park et al., 2008] indicate that 18.18% of their respondents report that a lack of manpower impacts the testing being performed - they view this as another significant problem. We recall that they do not distinguish what is meant by testing, therefore, whilst we find 62.16% of our respondents report a lack of people impacts software testing, overall (i.e. factoring in security testing and penetration testing) we find this becomes 35.65%.

A lack of budget is the next most common impact faced - across all activities - emphasising that SMEs are generally more resource restrained [Basri and O'Connor, 2010, Cruz-Cunha, 2010] and that V&V (particularly software testing) is costly in application (see Section 1.3.2). We then find a lack of people impacts security V&V and a lack of training and knowledge impacts software V&V. This further helps highlight that people appear more comfortable in performing software V&V, but that the activities themselves are impacted by a lack of people. In contrast, a lack of people is perhaps less of a concern when establishing and maturing a security V&V capability i.e. it is preferable to have a smaller number of highly-trained individuals capable of performing the activities. Lastly, we find commonality in a reported lack of tool support (we discuss this within Section 5.6.7.1).

Collectively, these results indicate that the software V&V activities have reached a higher degree of maturity, and organisational support, than the security-focused ones. Specifically, a lack of training and knowledge is more likely to impact the security-focused activities (predominately the test-based activities, but also almost a third of those performing security-focused design-based activities) and a lack of people the software V&V activities (most notably, software testing). A lack of budget and time is likely to effect both sets of activities equally, with penetration testing being somewhat less impacted by a lack of time (which we attribute to the reliance on outsourcing, see Section 5.6.2).

5.6.5.2 Undesired Social Factors

In the majority of organisations (some 70.27%), at least one activity practiced is impacted by a social factor, within the remainder (29.73%), no social factors negatively influence any of the activities practiced. Significantly, within 24.32% of the organisations, there are problems with all of the activities practiced.

The primary social factor, impacting all activities, is their being viewed as dispensable, see Figure 5.27. The only exception to this is software testing, where a lack of management support is the primary issue. V&V activities being deemed dispensable has previously been reported [Rodrigues et al., 2010] and is supported by comments from our respondents. For example, one Development Manager [O:BN-I:AC] particularly emphasises this: "*[w]hilst Testing is paramount, small teams have to make a call day to day on what they focus on. testing normally is the one where "we'll come back to it*"". However, unlike previous studies, we confirm this impacts the security-focused activities, which is perturbing given the impact of vulnerabilities (see Section 1.3.4). Notably, we cannot attribute this to an ignorance of the impact of security-related issues since several respondents actively showed an appreciation for security e.g. a Customer Support Manager [O:RV-I:LH] indicated they are "*very security-conscious*", whereas a Test Manager [O:RW-I:RL] stated: "*I think that security is incredibly important in this day-and age*".

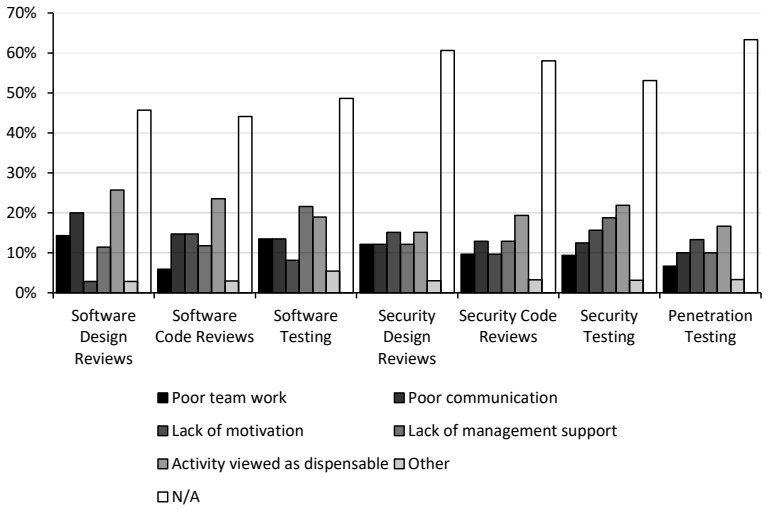


Figure 5.27: Undesired social factors

Unlike the resourcing constraints, there are less obvious themes with the social factors reflecting a divide between the software and security-focused activities. However, one significant theme which does exist - and potentially helps explain why respondents

may acknowledge the importance of security but, in turn, view the activities entrusted to find vulnerabilities as dispensable - is that the security-focused activities are less likely to receive management support. Specifically, the second most prevalent issue affecting security V&V is a lack of management support. In contrast, poor communication impacts software V&V. Further compounding the situation, the security-focused activities are also more likely to suffer through a lack of interactor motivation. Significantly, this is one of the primary reasons impacting security V&V but is the least reported undesired social factor influencing software V&V. We recall that a lack of motivation has a strong impact on whether an activity is performed, as well as its effectuality e.g. impacting test [Runeson, 2006] and review-based activities [Kollanus and Koskinen, 2006]. Notably, these studies focus on software V&V activities. However, we, like [Perry, 2006], believe that a lack of management support can result in engineers losing both motivation and interest in a task. Therefore, as a lack of management support is one of the primary social factors influencing security V&V, it holds that a resulting lack of motivation would also feature prominently.

Whereas other themes exist, such as poor communication being more likely to impact review-based activities than test-based activities, they do not help emphasise, or identify, any divide between the software and the security-focused V&V activities. However, whilst resourcing constraints - which touch upon the technical - may, overall, be viewed as having a greater impact on V&V, it is clear that social factors also impact the activities and, as seen with the resourcing constraints, impact software and security V&V differently.

5.6.6 Existing Communication Forums

Within this section of our analysis we attempt to understand interactor communication ecology [Kling et al., 2003] and, like in [Urquhart and Currell, 2010], the medium of communication. We also identify the communication tools used by the interactors.

5.6.6.1 Levels of Communication Between Interactors

Whilst developers and test engineers are the primary interactors, with their involvement varying per activity (see Section 5.6.1), in terms of communicating the results of these activities, Figure 5.28 shows that developers are the most informed interactor. In comparison, whilst test engineers are informed of the test-based activities, results from review-based activities are not typically communicated and, when they are, they are more likely to include design reviews than code reviews. These findings strengthen the view that activities at the code level are very much within the domain of the devel-

oper, showing that the interactors possess different skill sets (thus echoing [Dhaliwal et al., 2011]). However, aside from helping confirm the differences between interactors, it potentially highlights an undesired state, since: code reviews help communicate key information, permitting familiarisation with the "inner workings, design tradeoffs, and open issues with areas of the code" [Coram and Bohner, 2005, p. 368], thereby helping test engineers focus their testing. By not possessing this level of understanding (i.e. ignoring the internal workings of the product and treating it as a black-box), the ability to successfully detect vulnerabilities is reduced [Thompson et al., 2002, Potter and McGraw, 2004]. Thus whilst this does not impact generic test-based activities, it does impact those which are security focused.

Unsurprisingly, test engineers are generally better informed than developers when it comes to test-based activities, with the one exception being penetration testing. However, this is explained by the significant reliance on outsourcing (see Figure 5.12) where, in the author's experience, the relationship between the outsourced penetration test agency would be managed by development, as opposed to test. The reliance on outsourcing is probably also responsible for the decrease in verbal communication for this activity (which, on average, is approximately half that of the other activities, see Figure 5.29).

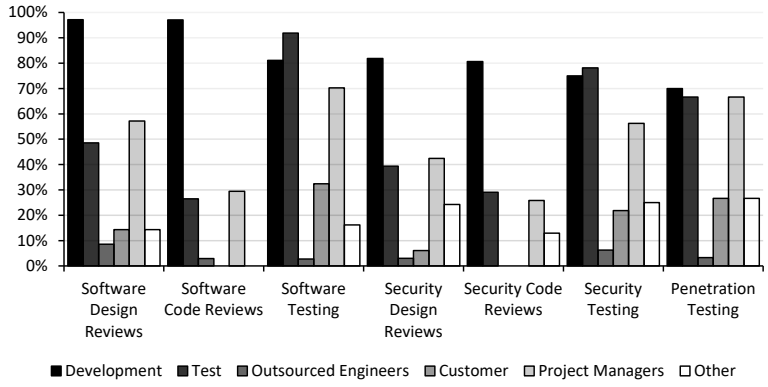


Figure 5.28: Interactors receiving activity related communication

Other types of interactor e.g. product management, support and senior management are also involved in the communication flows surrounding the activities but, notably, not in terms of code reviews. When customers were provided results, these typically concerned the test-based activities and, to a lesser extent, design reviews. Notably, for both, a slight decrease is found when comparing security-focused activities with software V&V activities (see Figure 5.28). We attribute this to security-related findings being considered sensitive (see Section 5.2). Similar results were found when examining

project managers - although, in all instances, they are generally more likely to be informed than customers. Also, unlike customers, they see the results from code reviews (both software and security-focused). This may just be an instance of project managers keeping track of technical tasks, as found by [Kautz and Thaysen, 2001] when studying a small software organisation. Or, it may be an instance of employees wearing multiple hats [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018], coupled with the view that project managers may not exist within smaller software organisations, with elements of project management being considered detrimental in such a context [Fayad et al., 2000] - thus implying a software engineer, or a former software engineer, may perform the project manager role (an approach observed within the author's working experience).

5.6.6.2 Types of Communication Between Interactors

Examining the type of communication between interactors, Figure 5.29 shows that test-based activities are more likely to have supporting formal documentation communicating the results of the activities than review-based activities. This can be attributed in part to the significant levels of outsourcing found in terms of penetration testing (see Figure 5.12, where formal reports from outsourced agencies would be expected) and that the test tools employed typically offer the ability to auto-generate sets of results (see Section 5.6.3). For example, within one medium-sized organisation [O:WS], the only formal documentation concerns penetration testing, an activity which the organisation outsources. The organisation's QA Director [O:WS-I:WE] also indicated: the "*[r]esults of automated testing are a factor in all releases. The automation runs either after a deployment to a staging server or nightly. There is no formal document but a process to review runs and correct any failures or regressions*". In contrast, verbal communication is the primary means of communicating when performing software code reviews which supports, and extends, the context of the findings within [LaToza et al., 2006] (their study was conducted within Microsoft) i.e. developers spend more time discussing code-related information face-to-face.

Typically, a variety of communication mechanisms are used per activity. For example, one Test Engineer [O:SV-I:TD] indicated that communication concerning security-focused review-based activities would occur verbally during meetings and within the tool where the issue was originally raised. They also indicated that a combination of informal documentation and verbal communication is used during software code reviews. Additionally, their organisation uses informal documentation for their security-focused test-based activities - with the notable proviso that they are "*lightweight, unless customer req'd*" - which highlights that the type of communication can vary dependent on the type of interactor involved. Further, it shows a slight difference in how software

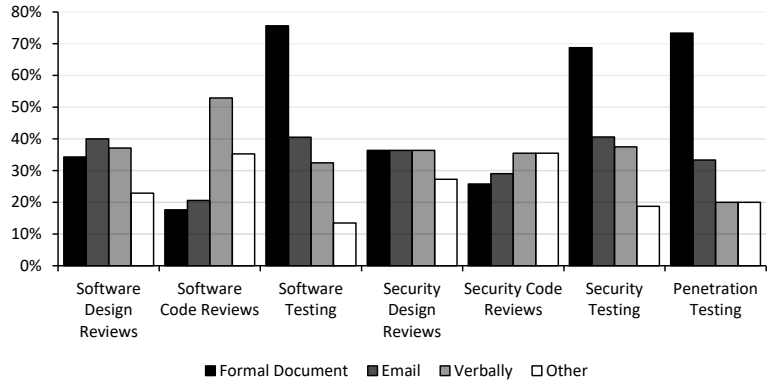


Figure 5.29: Type of communication

and security-focused activities are communicated. However, more generally, the findings show little difference between software and security V&V when it comes to the type of communication involved.

5.6.6.3 Tools Supporting Communication Between Interactors

Some of the tools directly supporting the V&V activities (see Section 5.6.3) are also used for interactor communication. For example, one Scrum Master and Lead QA Engineer [O:VS-I:GH] reported that conversations, concerning all software and security V&V activities practiced, are logged within Microsoft's *Team Foundation Server*. Similarly, a Security and Crypto Specialist [O:TN-I:AE] indicated that discussions would occur within *Coverity* when performing software and security-focused code reviews. We also found that the tools supporting software and security-focused test-based activities were used to disseminate auto-generated reports. These examples highlight that many of the tools supporting communication between interactors are equally applicable to both software and security V&V.

Unfortunately, the presence of these tools does not ensure strong levels of communication between interactors in terms of the security-focused activities; neither does it help ensure that the activities are performed. For example, although a variety of collaboration software, such as *Jira*, is used to capture and assist discussions between interactors, it is very notable, on exploring such examples, that differences between the software and security-focused activities are highlighted. For instance, a Test Manager [O:MC-I:AP] indicated that the results of software design reviews are "*logged in JIRA*" but, regarding security design reviews, stated: "*if done it will be logged in JIRA*". This respondent also indicated that *Jira* was used for software code reviews, but for security code reviews indicated "*don't know*". This confirms the finding within Section 5.5 -

namely, that security V&V activities are not performed as frequently - it also highlights that the presence or absence of a supporting tool is not what prohibits some activities from being performed.

Further, and aside from the tools identified in Section 5.6.3 (which directly support the V&V activities), we found use of code management systems, defect tracking systems, email, and project management software all being used to support the communication surrounding V&V e.g. in terms of code management systems, a Head of Quality Assurance [O:AV-I:JG] indicated: "*[d]etails [are] included in check in notes*". There was only one instance of project management software being used, supporting the view that they interfere with communication [Beck, 1998].

5.6.7 Resource Flows

Since resource flows involve "following the money" [Kling et al., 2003, p. 58], we begin by examining how a typical project's budget is spent on V&V. As Figure 5.30 conveys, software testing incurs the most costs. There is significant support for this finding (see Section 1.3.2), increasing the confidence in the accuracy of our results i.e. we find an average level of expenditure of just over 37% (much of the existing literature, which is not positioned solely within an SME context, indicate values approaching 50%). Similarly, we find comparable values to those reported in [Fagan, 1986] regarding software design and code review activities (i.e. ∼15%).

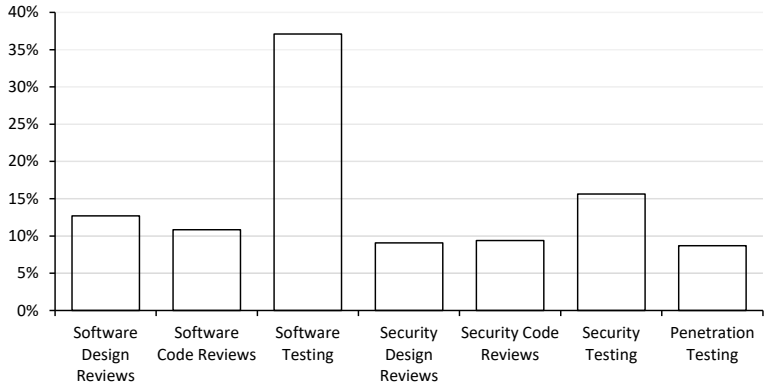


Figure 5.30: Average expenditure on a typical project

However, Figure 5.31 reflects the view that more investment is required across the activities, particularly regarding security V&V. Whilst this acknowledges the perceived value of security V&V, in corollary, it shows that respondents appear more satisfied with their existing levels of software V&V investment. Collectively, these results support the view that test-based activities are the primary approach to V&V [Vieira et al.,

2006, Ramler et al., 2006, Fernández-Sanz et al., 2009].

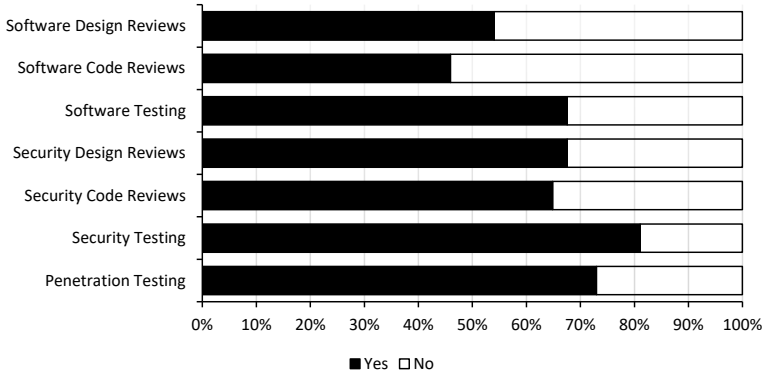


Figure 5.31: Is more investment required

We now examine where resource flows have had a direct or indirect impact on both the interactors and the activities.

5.6.7.1 Why Tools are not Utilised

It is clear that resource flows severely impact tool utilisation levels, with the primary reason being a lack of budget, see Figure 5.32. This is unsurprising since test automation and test tools are expensive [Sung and Paynter, 2006, Everett and McLeod, Jr., 2007, Vogel, 2011], with budget constraints known to impact test tool use [Ghag, 2008].

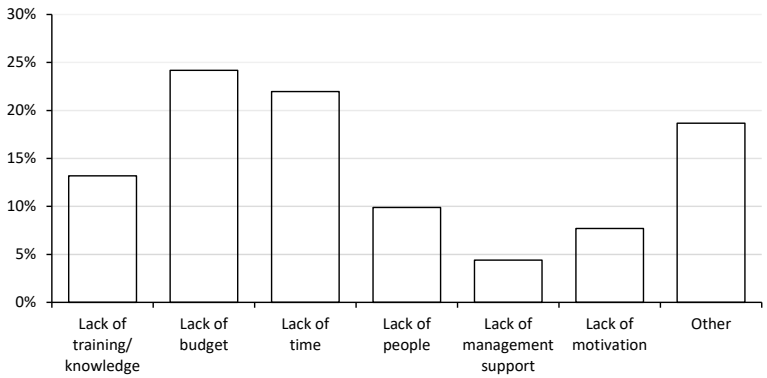


Figure 5.32: Why tools are not used

Lack of time is the next most frequently reported reason, followed by a lack of training or knowledge and then by a lack of people. Such constraints have been reported previously in a variety of contexts e.g. [Andersson and Runeson, 2002, Favre et al., 2003]. Notably, an often overlooked resource cost is the skill set required for the selected

test tool [Everett and McLeod, Jr., 2007], thereby implying more than a monetary resource flow constraint i.e. the cost of training, but also one of expertise. As small organisations have limited time and financial resources [Ward et al., 2001], this can result in limited training [Basri and O'Connor, 2010]. Whilst [Ward et al., 2001, Basri and O'Connor, 2010] focus upon small software organisations, we find that these constraints are not only present in software SMEs, but that they also directly impact the V&V activities. Notably, the majority of the reasons presented in Figure 5.32 all have a resource flow origin, with a lack of management support, and a lack of motivation, not appearing as such significant factors in preventing tool use (and, where they do appear, their split is fairly even across the software and the security-focused activities).

In some instances, within small and medium-sized organisations, there is no perceived need for a tool. For example, a Head of Development [O:AD-I:MD] reports: "*[w]e carry out manual reviews of Design documentation and see no need for tools to control this process*". Similar is echoed by a QA Director [O:WS-I:WE]: "*[a]t present the organisation sees no need for any tool in this area*". Others indicate preference for manually performing the activities e.g. a Test Engineer [O:SV-I:TD] succinctly notes "*[m]anual inspection preferred*" in terms of design review activities; and, a Head of Applications Development [O:OS-I:ST], indicates: "*we have tools that manage the process/workflow but the actual review is a manual activity*". It is also apparent, for the security-focused review-based activities, that a light-touch is taken towards tool use. This is evidenced by a Test and Support Manager [O:US-I:GJ] indicating they "*just have to confirm generic guidelines are followed during code design and review*". Therefore, tools are not employed. Notably, several respondents express views supporting one of the findings in Section 5.6.2, namely, they are more likely to outsource their security V&V needs (which is the case in small and medium-sized organisations). For example, for security code reviews, a Test Manager [O:MC-I:AP] indicates that "*this might be covered by any pen test activity we outsource*". Regarding penetration testing, where we have already seen significant use of outsourcing, a Head of Engineering [O:KD-I:MF] states that "*[w]e outsource penetration testing to a third party*", with a Chief Technology Officer [O:DS-I:SW] indicating that "*[w]e don't do any pen testing - the customers do that if they wish, with our support*". These examples also show a reliance on the secondary human interactors (see Section 5.6.2) when performing some of the security V&V activities.

It is also notable that small organisations are less likely to employ tools than medium-sized organisations, see Figure 5.33. This is applicable to all activities and further supports the view that smaller software organisations are more resource con-

strained.

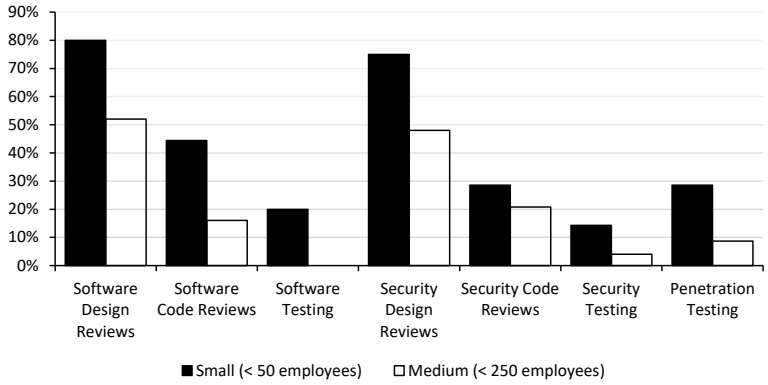


Figure 5.33: Tools not used by organisation size

5.6.7.2 The Use of Automation

Acknowledging the obvious value of automation (see Section 2.4), we observe that organisations investing in test automation can fail to realise their goals [Capgemini et al., 2009]. In general, test-based activities - both software and security-focused - are more likely to be automated than review-based activities, see Figure 5.34. Code review activities are then most likely to have some degree of automation, followed by design review activities. This strongly reflects the identified non-human interactors (see Section 5.6.3) i.e. test-based activities are more likely to have supporting tools e.g. test frameworks such as *NUnit*, thereby implying automation. Likewise, but to a lesser extent, we found static and dynamic code analysis tools being referenced, for example, *Coverity*. Thus, these results help support one another.

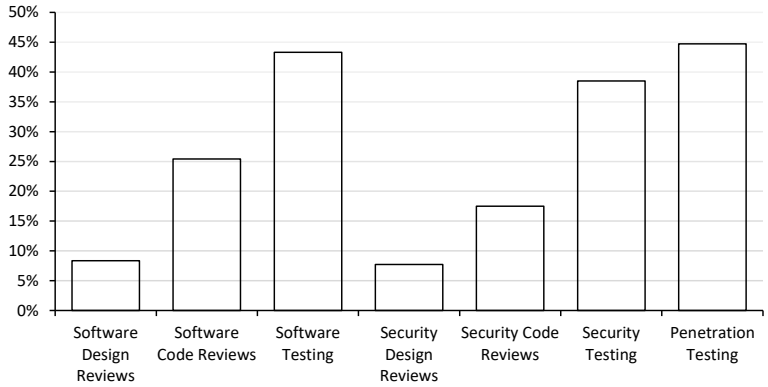


Figure 5.34: Average automation levels

Generally, medium-sized organisations are more likely to have reached a higher level of automation on average (see Figure 5.35), which supports the view that automation is expensive [Everett and McLeod, Jr., 2007], thereby potentially reducing the levels found within smaller organisations.

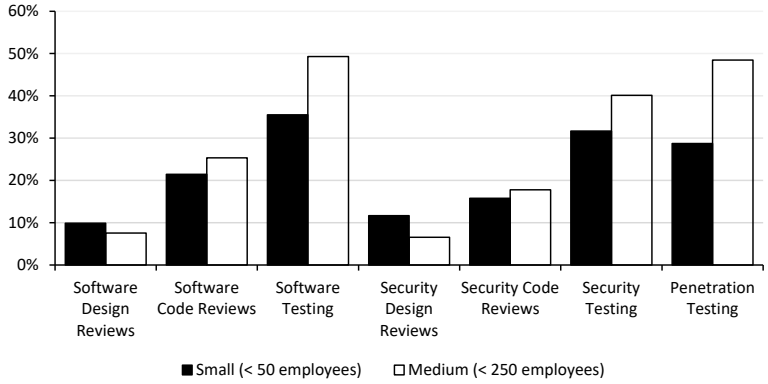


Figure 5.35: Average automation levels by organisation size

5.6.7.3 Afforded Levels of Training

We have acknowledged that software engineering is a complex, socio-technical activity [Sawyer, 2004]; that such complexity impacts V&V [Brooks, Jr., 1987, Xiao et al., 2007] and that finding vulnerabilities is hard [Austin and Williams, 2011] (see Chapter 3). To perform effective security V&V, developers and test engineers must supplement their existing knowledge with an understanding of software security [Kreeger, 2009, Felderer et al., 2011, Austin and Williams, 2011]. As highlighted in [Kreeger, 2009], graduates entering the workforce are ill-prepared for performing security testing and, within Section 2.6, we touched upon the relationship between an individual's knowledge and their ability to successfully perform V&V. Compounding the situation further, both software engineering and software security are ever-evolving disciplines, thus any formalised knowledge becomes quickly outdated [Bryant, 2005].

Having summarised the importance of possessing, and developing, knowledge so as to perform V&V effectively, we also highlight the importance of training within an organisation. Thus, like [Meyer, 2007, Taylor-Smith, 2016], we consider expertise a resource flow. Notably, the level of training afforded to the respondents is limited, for example, Figure 5.36 shows very little training regarding review-based activities - both software and security-focused. In contrast, there is significantly more software testing training provided, with ∼70% of the organisations providing some form of training. This decreases noticeably when examining the security-focused test-based training sit-

uation and is more apparent in terms of penetration testing (which we attribute to the levels of outsourcing this activity attracts, see Figures 5.12 and 5.13).

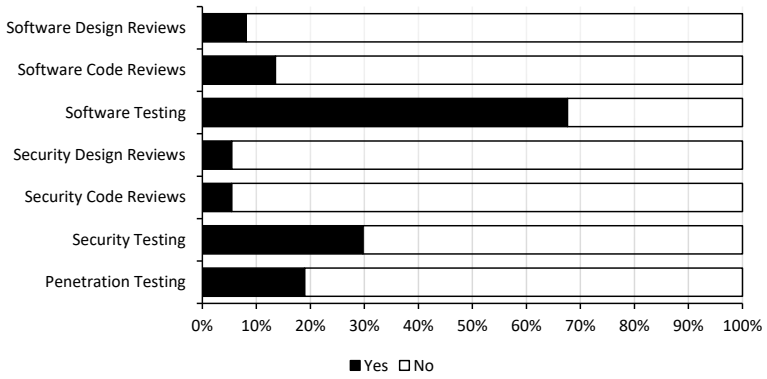


Figure 5.36: Levels of training per activity

Whilst the form the training received takes varies, see Figure 5.37, overall, organisations are more likely to support training in forms incurring the least direct costs i.e. through reading and mentoring (which, collectively, comprise almost 63% of the training received). This supports the findings of [Lethbridge, 2000, Sung and Paynter, 2006], who show the prevalence of on-the-job and self-directed learning in terms of V&V activities. Formal training courses, or conference attendance - which are likely to be more costly forms of training (formal training is often expensive and time-consuming [Rus and Lindvall, 2002]) - are less likely to be supported by the organisations. This is the case across all V&V activities, see Figure 5.38.

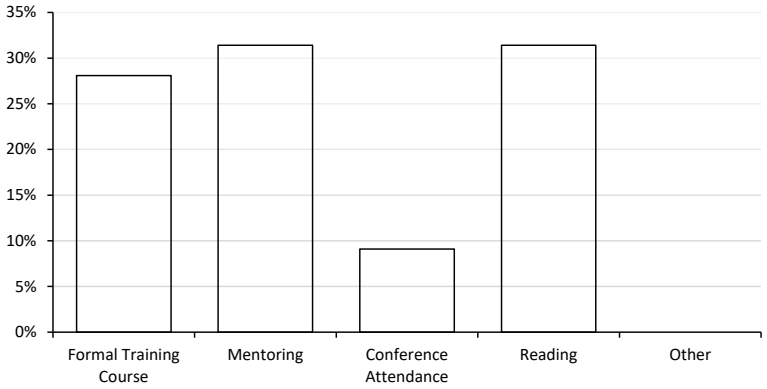


Figure 5.37: Forms of training supported by organisations

The regularity of the training received - over the past two years - is also limited (see Figure 5.39). That the majority is either on-the-job training, or through reading

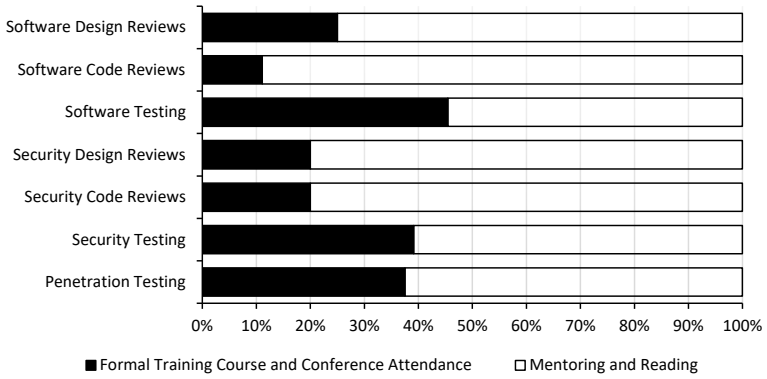


Figure 5.38: Forms of training per activity

a book, suggests this frequency is insufficient. Further compounding this, the majority of the training received occurred over 12 months ago (see Figure 5.40). Arguably, therefore, the training may now be out of date - especially when recalling the ever-evolving nature of software engineering, V&V and software security. In addition, if the training has not subsequently been put into practice, then the instruction received may have since been forgotten by the recipient. As noted by [Meehan and Richardson, 2002], in the context of software process training within small software organisations, any on-the-job learning must be supplemented with further support and guidance - thus implying a regularity which we do not see within Figure 5.39.

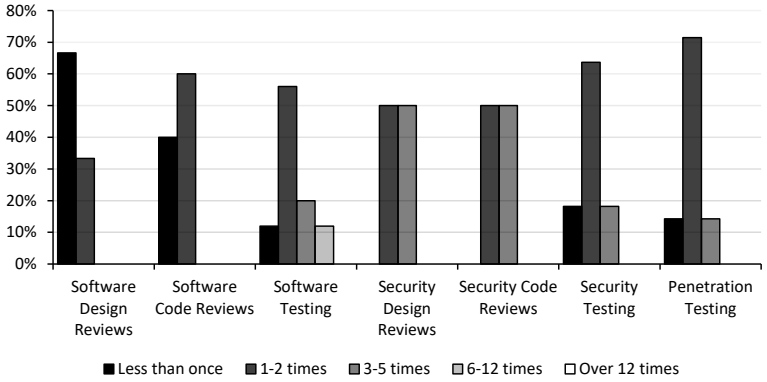


Figure 5.39: Regularity of training over last two years

In summary, the levels of training afforded by the organisations are constrained - being both limited in terms of opportunity and frequency - as well as being potentially out-of-date. These findings help support those of others e.g. that the resourcing constraints of SMEs can prevent sufficient investment in organisational development

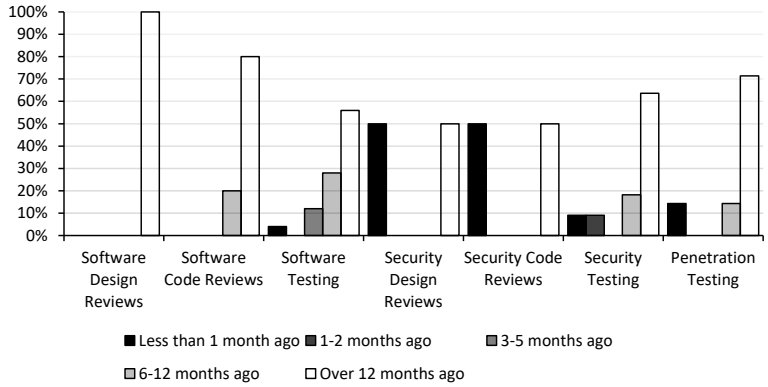


Figure 5.40: When training was last received

activities [Sitnikova et al., 2007], leading to limited training within small organisations [Basri and O'Connor, 2010]. Notably, we find the level of training offered does impact V&V within software SMEs (see Section 5.6.5).

We now address the second research objective.

5.7 Information Security Culture

Within this section we address the second research objective by determining whether information security culture influences security V&V within an organisation. To achieve this, we first establish whether an organisation has an information security culture. Notably, approximately 70% of the organisations report an information security culture, 22% that such a culture does not exist, with 8% unsure. Of those indicating an information security culture exists: approximately 4% indicate it is weak, 73% strong and 23% very strong. However, to support these findings - specifically, whether a culture exists or not - we analyse the responses received to the other questions within Section 2 of the survey instrument (see Section H.3). Utilising STIN to structure this analysis (see Section 3.3.6), we begin by identifying the human and non-human interactors involved with information security within an organisation (Sections 5.7.1 and 5.7.2). We then examine the motivations and incentives within an organisation which help establish an information security culture (Section 5.7.3), as well as the excluded interactors and the undesired interactions which impact this culture (Section 5.7.4). Finally, we examine the information security-related communication within an organisation and the resource flows surrounding an organisation's information security culture (Sections 5.7.5 and 5.7.6).

5.7.1 Human Interactors

Acknowledging the importance of an information security organisational structure [von Solms and von Solms, 2004], we find, within ∼78% of the organisations, at least one person responsible for information security. This is comprised as follows: organisations with a single individual responsible account for ∼30%, whereas a group of individuals, who share this responsibility, account for ∼48%. Whilst there were no instances where no one was responsible, almost 22% of the respondents were unsure (notably, 25% of these were small organisations, 75% medium). In contrast, 57% of the SMEs surveyed by [Dimopoulos et al., 2004] had someone responsible for information security (which can be contrasted with our ∼78%). Dimopoulos et al. also report a lower value for those unable to state whether someone was responsible or not (14%, as opposed to almost 22%). However, it is notable that 29% of the organisations they surveyed have no one responsible for information security. As their industry context is undefined, it is not possible to state why our results differ, however, as both study contexts involve SMEs, a possible explanation for our healthier picture may be due to the industry sector we have targetted i.e. software organisations with technology-focused employees that possibly have a higher-level of appreciation for information security. This is borne out by several comments from our respondents e.g. "*[b]eing in the security industry it is important for everyone to be aware of the threat landscape*" [O:AV-I:JG], "*I have a background in cryptography and information security products and am very security-conscious*" [O:RV-I:LH] and "*I think that security is incredibly important in this day-and age when dealing with customer data*" [O:RW-I:RL].

Where a single individual is responsible, 64% of these organisations report an information security culture, with the remainder (36%) stating no such culture is present. In contrast, where a group of individuals are responsible, 78% report an information security culture, 11% that there is no such culture, with 11% unsure. These results suggest that when a group of individuals are responsible for information security, organisations are more likely to have an information security culture, which reflects "security has to be a shared, not differentiated, responsibility among team members" [Majchrzak and Jarvenpaa, 2004, p. 11]. Regarding what constitutes a group, the majority of cases (just over 72%) comprise between 2-5 people, with almost 17% having between 6-10. The remainder indicated over 10 people, with one Test and Support Manager [O:US-I:GJ] indicating explicitly that "*everyone*" is responsible, which affirms the view that information security accountability must encompass all employees [von Solms and von Solms, 2004]. However, where a single individual is responsible, ∼64% of these hold other roles i.e. information security is not their sole responsibility (which further supports the view that employees within SMEs wear multiple hats [Murthy, 2009, Cruz-

Cunha, 2010, Linares et al., 2018]). Fifty-seven percent of these organisations report an information security culture, whereas 43% indicate there is not one present. Factoring in the "Do Not Know" responses, the percentage reporting an information security culture drops to 50%, with 50% indicating no such culture is present. This demonstrates the importance of someone being visibly responsible for information security within an organisation - which is emphasised when acknowledging that information security has moved beyond being a technical concern, having become a management one [von Solms, 2001a].

Where a group of individuals are responsible for information security, in the majority of cases (nearly 78%), they hold other roles. This, coupled with the instances where single individuals are responsible - and hold other roles - shows that dedicated roles for governing information security typically do not exist within SMEs. This echoes the view that dedicated security roles should be abolished [Majchrzak and Jarvenpaa, 2004]. In comparison to the instances of single individuals, almost 79% of these organisations report an information security culture, with only ∼14% indicating there is no security culture. This further shows that having more than one person responsible for information security (and not necessarily dedicated in role) will more likely result in employees feeling that an information security culture is present within their organisation. This emphasises that it is preferable to increase security knowledge, and awareness, throughout a small organisation, rather than just increasing the size of the information security department [Kajava et al., 2006]. However, whilst just over 72% of the organisations - where at least one person is known to be responsible for information security - hold other roles, we should not discount the 21% who indicate at least one individual is dedicated to information security within their organisations (whether acting individually, or as an individual representing a group of individuals), as in all of these instances an information security culture is reported to be in existence.

5.7.2 Non-Human Interactors

Recalling the importance of an information security policy [von Solms, 2001b], and that employees are often unaware of them [von Solms and von Solms, 2004], we find of the organisations possessing an information security policy (some 65%), 92% report an information security culture (the remainder do not). Conversely, when organisations do not possess a policy - as indicated by some 19% of the respondents - 71% of these do not have an information security culture and 29% are unsure. Of the 16% unware of whether a policy exists or not, 67% report an information security culture exists, 17% that one does not exist and 17% are unsure. It is thus clear that the presence of an information security policy is strongly linked to whether employees believe an

information security culture is present within their organisation, supporting the view of how fundamental such a policy is [von Solms, 2001b].

However, there is more to just being aware of a policies existence. Therefore, it is pleasing that 79% of those indicating an information security policy exists, also state they understand the policy, with 17% understanding it to some extent and only 4% indicating they do not understand the policy. Interestingly, when contrasting the respondent's own level of awareness and understanding of the information security policy with how they perceive the situation with their colleagues, we find 92% are aware of the policy, 4% are unaware and 4% were unsure. In terms of their colleagues understanding the policy, 67% understand the policy, 29% understand it to some extent and 4% do not know. This suggests that when an organisation possesses an information security policy it is generally acknowledged and understood to the same extent by employees throughout the organisation.

5.7.3 Incentives

Senior management should show visible adherence to an organisation's information security policy [Wood, 1997, Knapp et al., 2006, Thomson et al., 2006, Johnson and Goetz, 2007] as organisational culture alone is insufficient in fostering employee compliance [Hu et al., 2012]. It is thus promising that 79% of the respondents indicate their senior management follows their organisation's information security policy, with 17% indicating they do to some extent and only 4% indicating they do not know. Notably, within all organisations where senior management follow the information security policy, an information security culture exists (with 74% indicating it is strong, 26% very strong). However, where senior management are only partially seen to be following the policy, 50% indicate their organisation has an information security culture, 50% do not. Significantly, of those reporting an information security culture, 50% report that it is weak, 50% strong. This helps emphasise the importance of senior management being actively seen to support information security within an organisation.

That there is a relationship between an organisation's management and information security culture [Knapp et al., 2006, Kajava et al., 2006, Ruighaver et al., 2007], we examine how those deemed responsible for information security are positioned within an organisation. Where a single individual is responsible, and either reports directly to, or is, senior management (which is the case in 91% of the instances), 40% of the organisations have an information security culture, 60% do not. In the case where a group of individuals are responsible, and at least one member of this group reports directly to, or is, a senior manager (which is the case in just over 94%), only 6% indicate no culture is present, 12% do not know, whereas 82% indicate a culture is

present. Whilst these results indicate senior management are influential in establishing an information security culture, they suggest more strongly that a group of individuals, whether dedicated to information security or not, help promote such a culture. This could also support the view that "senior management isn't the biggest hindrance to better security" since "middle management might represent one of the largest challenges because they impact the organization daily" [Johnson and Goetz, 2007, p. 17].

The importance of security awareness, and employees being aware of their responsibilities, has previously been acknowledged [Williams, 2009]. We find that the majority of respondents are aware of their information security responsibilities (with 70% indicating they are, 22% to some extent, with only 8% being unaware). Notably, 74% of those asserting awareness (either fully or to some extent) indicate an information security culture exists. This can be contrasted to those indicating they are unaware of their responsibilities - whereupon, the organisations reporting an information security culture drops to 33%. It is thus clear that an employee's awareness of their information security responsibilities helps promote an information security culture. However, observing that policies should be enforced [von Solms and von Solms, 2004, Burns et al., 2006, von Solms and von Solms, 2009], one way of achieving this is to include these responsibilities within an employee's contract of employment (thus "employees consider compliance a condition of employment" [Wood, 1997, p. 18]). We find, where an employee's contract of employment incorporates these responsibilities (some 38%), that all of these organisations report an information security culture (factoring in those indicating that their contract of employment incorporates these responsibilities to some extent, results in 54% of the organisations, 90% of which report an information security culture - notably, 50% of the Welsh SMEs surveyed within [Burns et al., 2006] incorporate compliance with an information security policy). In contrast, where these responsibilities are not incorporated, we find that the likelihood of there being an information security culture drops to 36%, with 64% explicitly indicating there is no culture present.

Additional incentives, which influence compliance and security awareness, focus more at an organisation level as opposed to an individual e.g. achieving and maintaining various organisation-wide information security certifications is a strong incentive. As one Head of Applications Development [O:OS-I:ST] indicates: "*[w]e are ISO27001 certified. As part of this process we have a management team who are responsible for Information Security. This includes communication, reviewing security incidents, reviewing and sign-off on processes etc*". Similarly, a Head of Engineering [O:KD-I:MF] noted it is "*[b]ecause following ISO9001 and ISO27001 processes is important to the business and it is kept at the forefront of people's minds*". By meeting these external

certifications, some organisations have introduced mandatory, organisation-wide rulings. For example, a QA Director [O:WS-I:WE] informs: "*[w]e are now an ISO27001 company. Achieving this required full company awareness, handled by a specific team*", however, this was only effective as "*[i]t was compulsory for the entire company to understand and take part*". Similarly, and aside from just being aware of their own responsibilities, having an awareness of their organisation's wider information security responsibilities is also a significant factor in determining whether an information security culture is present. For example, of the 59% indicating awareness of their organisation's information security responsibilities, 86% report an information security culture. This can be contrasted to those who are only aware to some extent, as well as those who are not aware (which account for 35% and 5% respectively). In terms of the former, those reporting an information security culture drops to 46%, in terms of the latter, 50% report an information security culture.

Having identified the interactors and their incentives, we now identify the excluded actors and the undesired interactions.

5.7.4 Excluded Actors and Undesired Interactions

Although various attempts have been made to improve the level of security awareness within the organisations, it is apparent that not all were as effective as desired. Specifically, 41% of the respondents have, themselves, attempted to improve security awareness within their organisation (59% have not). We consider this number relatively high, and encouraging, based on the respondent profiles - namely, none are dedicated in role to ensuring organisational information security (see Section 5.4.2). Notably, 40% of those attempting to improve security awareness indicated their attempts were successful, 40% they were effective to some extent, 7% they were not effective and 13% were unsure whether their attempts had proven effective. However, within the context of the wider organisation, 70% of the respondents reported attempts to improve the level of security awareness, 24% were unaware of any such attempts and 5% indicate that no attempts have been made. Of the attempts made by others, 46% have proven effective, 35% were effective to some extent, 4% were not effective and 15% were unsure.

These findings suggest undesired interactions are influencing the effectiveness of attempts to improve the level of information security awareness. For example, even when an information security governance structure is present, communication barriers can inhibit progress. As one Test Manager [O:RW-I:RL] found when attempting to raise security awareness: "*[a]s a tester I execute the vulnerability scanning utility against a range of environments out-of-band. I communicate the results out*". However, when communicating this information within the organisation, and more generally, they indi-

cate the surrounding communication is somewhat lacking: "*[i]nfrastructure have talked to me at times about it... along with the CSO (Chief Security Officer). Its difficult to assess this as I am not part of the Infrastructure team which has involvement with the customer environment scanning*". They also observe, again in the context of conducting vulnerability scanning: "*I think that not all recipients can interpret the scan results so it goes a little over [their] head... plus, they have other tasks which they feel are more pressing*". This highlights two additional problems, namely: not all interactors are equally fluent in the topic of discussion (emphasising the impact of its complexity, see [Wang and Guo, 2009]) and, possibly because of this, other tasks are deemed higher priority (which echoes a Senior QA Manager [O:CR-I:MC]: "*[w]e can not always apply the changes required with project timeframes unless critical*").

Such undesired interactions are reported by other respondents. For example, a Chief Technology Officer [O:CS-I:GB] questions: "*do people really understand what they heard*" when attempting to improve security awareness. Specifically, they observe it is "*[d]ifficult to tell how much someone has learned/understood. Would be better to have a more positive feedback mechanism*". Failures in communication also help highlight instances of potentially excluded actors. As observed, not all organisations have employees dedicated to information security (see Section 5.7.1). This can limit the effectiveness of information security-related communication since, as another Chief Technology Officer [O:DS-I:SW] observes: "*[t]he security person gets to see/talk to everyone*". Without such a person, or group, acting as a focal point, it is apparent, at least within some of the organisations, that information security-related discussions may not be as effective as desired. However, even when such a group exists - if the level of communication is not appropriately managed - the results may be less effective. As one Test Manager [O:MC-I:AP] states: "*[w]e have a compliance team that enforce the security policy and standards*", but who then acknowledges that for such activities to be more effective it "*needs to be more frequent*".

We also find a lack of support being reported. For example, a Scrum Master and Lead QA Engineer [O:VS-I:GH] observes: "*[t]rying to educate users and encourage management to provide more resource for security*" has not been as effective as it could have been since "*[t]hey are still 'thinking' about it!*". This conveys not only a tone of exasperation, but also emphasises that when there is a failure to gain management support, the task of acquiring the relevant information security resources is further compounded. A Development Manager [O:BN-I:AC] also highlights the difficulty in acquiring support: "*I think security only raises its head when things go wrong. maybe some head burying in the sand, because its investment in intangible software*". This reflects the need to become more proactive with security [Johnson and Goetz, 2007].

Aside from respondents reporting difficulty in acquiring management support, we find only one instance where the lack of support - in terms of raising security awareness - appears a more pervasive problem e.g. as one Test Manager [O:RW-I:RL] observes: "*[o]ther than a dev or two plus the roles I have already mentioned... no one else seems to raise awareness of this*".

5.7.5 Existing Communication Forums

It is important, when an information security policy exists, that updates to the policy are effectively communicated to an organisation's employees (without this, employees will likely remain unaware of its existence [von Solms and von Solms, 2004]). Notably, 88% of the organisations notify their employees when their policy is updated (e.g. one Head of Engineering [O:KD-I:MF] notes: "*[o]ur IT manager regularly updates the process and communicates this*"), the remainder do not. Overwhelmingly, email is used to communicate these updates (within 95% of the organisations), with verbal communication being the next most common (accounting for 29%). We also find, albeit at much lower levels of utilisation, that organisations utilise their company intranet or HR systems to provide policy notifications and updates, with some also employing flyers and posters.

In terms of raising information security awareness in general, a variety of communication mechanisms are used - which is important, since key to improving security awareness is "varying the delivery mechanisms, to keep everyone interested" [Kruger and Kearney, 2006, p. 290]. That various communication mechanisms are employed is evident by the variety of respondent comments e.g. one Head of Quality Assurance [O:AV-I:JG] has conducted several "*[s]ecurity sessions and demonstrations*" within their organisation; a Product Development Manager [O:MF-I:TE] has captured security as an "*[a]genda item at [their] Department Meeting*"; a Customer Support Manager [O:RV-I:LH] reports they have "*[c]irculated articles on security advice, hacks, security advisories, etc*"; a Chief Technology Officer [O:CS-I:GB] indicates they have provided a "*[v]ariety of talks, email and internal articles*" within their organisation; whilst others indicate internal talks are given and that the topic is "*[r]aised at briefings*" [O:CB-I:PT].

5.7.6 Resource Flows

Although [Ruighaver et al., 2007] state that too many organisations offer limited information security training on employee induction, the organisational and industrial contexts on which this assertion is based are not elaborated. However, we find this

also holds true within our study context, since only 38% of the respondents received information security training during their induction, 51% did not receive any training and 11% are unable to recall. Notably, there is a general consensus that subsequent training is required, on a continual basis [Kruger and Kearney, 2006, Ruighaver et al., 2007]. However, aside from induction training, only 27% of the respondents received additional information security-related training (65% of the respondents have not received any additional training, 8% are unable to recall). Interestingly, of those who received induction training, 86% believe an information security culture exists (with only 7% indicating such a culture does not exist, with 7% unsure). This drops to 58% when not receiving any induction training (with an increase to 37% of those reporting no information security culture, with 5% unsure).

Significantly, in all instances where subsequent training has been received, an information security culture is reported to exist. Where subsequent training was received, 60% indicated a single extra training session, 40% two sessions. Notably, no one indicated higher than this even when considering the average respondent tenure (see Section 5.4.2). Collectively, these results support the importance of training, and the periodic repeating of refresher training, within an organisation.

Having addressed the research objectives, we now provide some conclusions and reflect upon this phase of data collection.

5.8 Conclusions and Reflections

Within this phase of data collection we employed a survey to address the identified research objectives. We begin by summarising the findings for the first and third research objectives.

5.8.1 Software and Security V&V Practice

The first and third research objectives involved examining how security V&V is practiced, supported, and perceived, within UK-based software SMEs, as well as establishing whether they can possess a mature security V&V practice without a correspondingly mature software V&V practice. We structure our conclusions in terms of the interactors and their incentives, and then in terms of the relationships which exist within the socio-technical network.

5.8.1.1 Interactor Analysis Summary

We found a fairly diverse set of human interactors across the various V&V activities, with many of these interacting with one another on the same activities e.g. developers

and test engineers both performing security testing. Further, we observed diversity regarding the tools and processes used.

By identifying these interactors, we highlight:

- There was limited evidence that software SMEs make much recourse to internal security expertise. Additionally, there was a notable absence of roles dedicated to performing security V&V (see Section 5.6.2).
- Whilst there was a desire to further employ outsourcing in general, this was more marked in terms of security V&V (see Section 5.6.2).
- Software V&V activities were more likely to have tool support (see Section 5.6.3.1). Respondents were also more able to discuss the tools used within the context of such activities.
- There were more likely to be processes governing the software V&V activities than the corresponding security-focused activities (see Section 5.6.3.2). Once again, respondents were more confident indicating the situation within a software V&V context.

When the above is coupled with the findings in Section 5.5, namely: security V&V is performed less frequently than software V&V, with respondents having greater difficultly in elaborating upon the security-focused activities being performed within their organisations, we conclude that SMEs are much more active, and indeed confident, in their engagement with software V&V than security V&V. Specifically, software SMEs have greater confidence in their ability to perform, and own, software V&V activities. We also find that these activities are better supported, and managed, within SMEs, indicating they have reached a higher degree of maturity within the organisations surveyed. However, with this in mind, we observe that a variety of incentives exist within the V&V socio-technical network which help motivate the interactors. Specifically, we highlight that:

- The primary motivator in performing any V&V activity was a desire to improve software quality. However, this was more prevalent in terms of the software V&V activities than the corresponding security-focused activities. We ascribe this to interactors not affording importance to security as a quality attribute.
- Acknowledging and adhering to a governing process was the second most common motivator. Supporting the results presented in Section 5.6.3.2, this is more pronounced in terms of software V&V than security V&V, which is further evidence that these activities have reached a higher level of maturity.

- There is little indication that interactors are motivated by a sense of fun when performing any V&V activity. Conversely, there is little indication they are motivated solely to "tick a box". However, in terms of the latter, the majority of instances concerned the security-focused activities.
- Customers appear to be a stronger motivating force in terms of test-based activities - both software and security-focused - than review-based activities.

Collectively, aside from a desire to improve software quality, interactors appear more motivated by external influences i.e. mandating processes, instruction from management and requests from customers, than through intrinsic factors such as deriving enjoyment from the activity itself. That the most prevalent incentive is a desire to improve software quality is supported by the existing literature on motivation in software engineering (see Section 5.6.4.1). However, we confirm this within an SME and a V&V context. Notably, the software V&V activities are more likely to be motivated by this intrinsic factor than the security-focused activities, which is a clear indication that security, as a quality attribute, is not afforded, or perceived by the interactors as having, as much worth as other quality attributes.

5.8.1.2 Network Relationship Analysis Summary

It is apparent that a variety of undesired interactions, both social and technical, exist and that some of the identified interactors either are, or should be, excluded from certain activities. Specifically, we found:

- That the activities, as practiced within the vast majority of organisations surveyed, suffer due to resourcing constraints and/or social factors (see Section 5.6.5). Notably, the software V&V activities are more likely to be impacted by a lack of people, whereas the security-focused activities are more likely to be impacted by a lack of training and knowledge. Additionally, the security-focused activities are less likely to receive management support, with the software V&V activities more likely to suffer from poor communication.
- The security-focused activities are more likely to involve a subset of the identified interactor groups (see Section 5.6.5). This carries the implication that the software V&V activities are more inclusive in terms of the interactors identified. Notably, there are more reported instances of wanting to exclude interactors in terms of the security-focused activities.

The security-focused activities are also more likely to suffer through a lack of interactor motivation. This is somewhat unsurprising given that the security-focused

activities are more likely to be impacted by a lack of management support - thus, by extension, the level of motivation exhibited by the interactors themselves will likely be weakened. In turn, this is likely to result in the activities being performed less effectively. This is liable to result in a vicious cycle, whereupon, an organisation is not witness to any significant benefits - as the activities are being practiced in a sub-optimal manner - therefore, there is not such a strong incentive to invest and actively support the activities. We find evidence to support the view that only by failing in terms of software security is an organisation more likely to become actively invested in such activities. As stated by a Development Manager [O:BN-I:AC]: "*I think security only raises its head when things go wrong*"; similarly, a Head of Testing [O:WK-I:RR] indicates: "*I would say we generally have the (in my opinion, wrong) opinion that if customers want to test our products for security issues they can, and we generally only perform security testing when forced. I am attempting to change the culture within the organisation to bring the importance of security testing up the scale. We have had some incidents lately with customers that are starting to support my case*". As the costs of such failure can be high (see Section 1.3.4), this is an expensive lesson to learn.

Continuing with the theme of investment, and that of resource flows in general, we observe:

- The average level of expenditure, per activity, is higher in terms of the software V&V activities than the corresponding security-focused activities (see Section 5.6.7).
- Respondents were more likely to indicate that additional investment was required regarding the security-focused activities than the software V&V activities (see Section 5.6.7).

Whilst the latter finding provides some acknowledgment that the security-focused activities require more investment, such investment needs to be carefully directed. For example:

- Whilst a lack of budget is the primary reason for why tools are not used, it is the primary reason for why both software and security-focused tools are not used (see Section 5.6.7).
- The tools supporting communication between the interactors can, and are, applied to both the software and the security-focused activities (see Section 5.6.6).
- The average levels of automation (we recall automation is considered expensive, see Section 1.3.2), per activity, are comparable when contrasting the software and corresponding security-focused activities (see Section 5.6.7).

However, investment in improving the levels of employee knowledge - particularly with regards to security V&V - could make a significant difference. Notably, the security-focused activities are more likely to suffer from a lack of interactor training and knowledge than the software V&V activities, with this being the most common reason why an organisation never performs a security V&V activity. Generally, the level of V&V training received within the surveyed SMEs was limited, but it was more so in terms of the security-focused activities. As found, this has a greater detrimental impact on the security-focused activities.

In summary, the level of interactor interaction surrounding the security-focused activities is less inclusive - which reflects the sensitive nature of security-related discussions (within one organisation the author has worked for, vulnerabilities would typically remain "hidden" to the majority of the engineering team until fixed). However, the communication between interactors is generally consistent across the software and security-focused activities regarding both the mechanisms and tools employed. This is probably attributable to the fact that such communication mechanisms, and supporting tools, have already been established within the organisations and, regardless of activity and focus of communication, are deemed equally applicable. It is, however, significant that the lack of security-focused training and knowledge is one of the primary factors impacting the security-focused activities. This further emphasises that security V&V encompasses a complex set of activities, requiring a blending of knowledge, across a variety of domains. Unfortunately, whilst there is a recognised need for further investment in security V&V within the organisations, it is clear that this has yet to result in the provision of training. In general, training was found to be rather limited, and whilst this is supported by the existing literature regarding SMEs (e.g. [Stokes, 2001]), we are able to confirm that this extends to encompass all V&V activities. Further compounding the situation, the security-focused activities are less likely to receive management support and are more likely to suffer through a lack of interactor motivation.

Based upon the analysis performed, we thus conclude that the software V&V activities have reached a higher level of maturity. This is evident in terms of resourcing, the application of the activities themselves, as well as their acceptance within the organisations which is demonstrated by the greater level of organisational support being afforded and the surrounding levels of awareness.

We now present our conclusions concerning the second research objective.

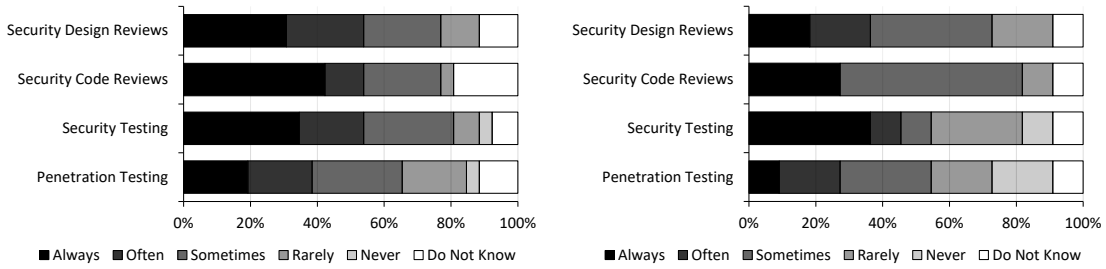
5.8.2 Information Security Culture and V&V Practice

Following the analysis performed (see Section 5.7), we find that the presence of an information security culture - as indicated by the respondents - is consistent with their responses to the other questions within Section 2 of the survey instrument (see Appendix H). Therefore, we can now address the second research objective which seeks to determine the influence an information security culture has on the security V&V activities being performed within UK-based software SMEs.

We begin by examining the impact an information security culture has on the frequency with which the various V&V activities are performed.

5.8.2.1 Activity Frequency

Figure 5.41 shows that all security V&V activities are practiced with more regularity within organisations reporting an information security culture than those without. In corollary, the absence of such a culture corresponds with a reduction in how often the activities are performed (i.e. there are more instances of "Rarely" and "Never" being reported than those with an information security culture).

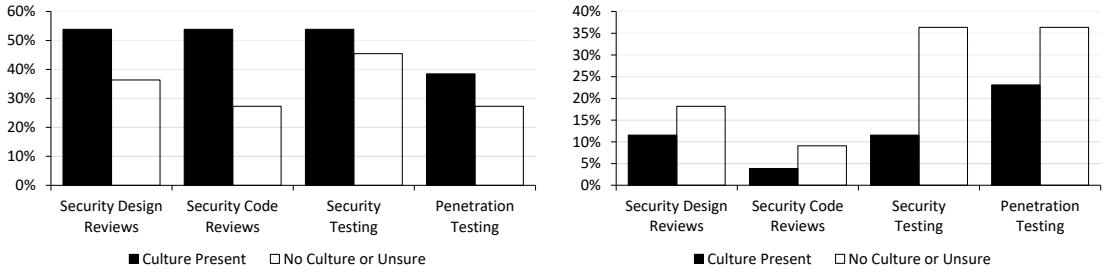


(a) Culture present (b) No culture or unsure

Figure 5.41: Influence of information security culture on activities

Figure 5.42 further highlights the influence an information security culture has on the frequency with which the individual security V&V activities are performed. Specifically, examining the activities which are performed regularly ("Always" and "Often") and those that are performed less regularly ("Rarely" and "Never"), it is apparent that the activities are performed more regularly within organisations with an information security culture.

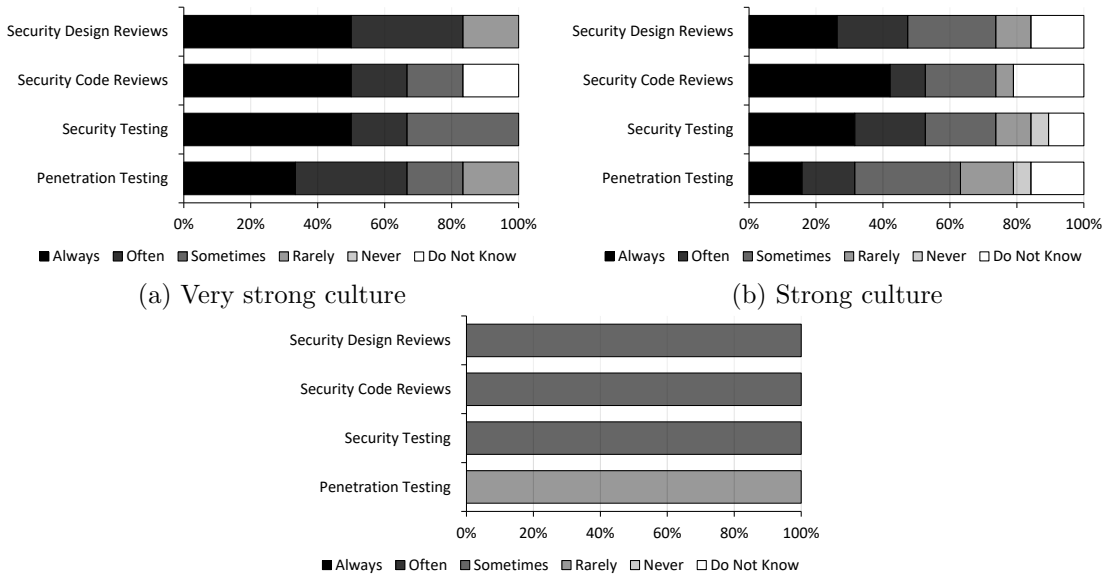
We also find that the strength of the culture further influences the frequency with which the activities are performed. As Figure 5.43 shows, a stronger culture results in an increased frequency in how often the activities are performed and, correspondingly, a weaker culture results in the activities being performed less often. The impact that



(a) "Always" and "Often" (b) "Rarely" and "Never"

Figure 5.42: Influence of information security culture on activity frequency

the strength of an information security culture has on the activities being performed regularly i.e. "Always" and "Often", is reflected in Figure 5.44.



(a) Very strong culture (b) Strong culture

(c) Weak culture

Figure 5.43: Influence of information security culture strength on activities

Examining whether an information security culture also impacts the software V&V activities, we find, when performed, that they appear to be less influenced by whether an information security culture exists or not (particularly so with regards to software testing), see Figure 5.45. This further supports the view that there is a genuine relationship between an organisation possessing an information security culture and the activities associated with security V&V. Specifically, the presence of an information security culture does influence the level of security V&V being practiced within SMEs.

We now examine the influence an information security culture has on the non-human interactors within the V&V socio-technical network, as well as the communication and

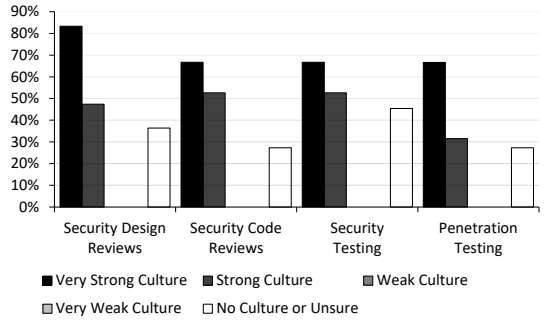
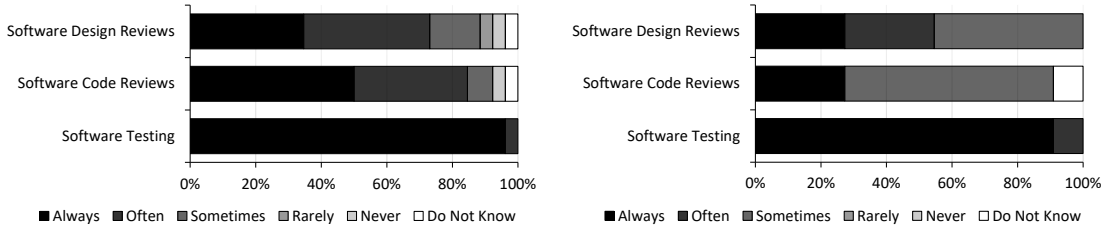


Figure 5.44: Influence of information security culture strength on activities performed regularly



(a) Culture present

(b) No culture or unsure

Figure 5.45: Influence of information security culture on software V&V

resource flows surrounding the security V&V activities.

5.8.2.2 Non-Human Interactors

Whilst organisations with an information security culture are more likely to have an information security policy (see Section 5.7.2), Figure 5.46 shows that such organisations are also more likely to have processes governing the various security V&V activities. We recall that many maturity models, e.g. [CMMI Product Team, 2010, Kollanus, 2011], view the presence of a governing process as an indication of having reached a higher level of maturity.

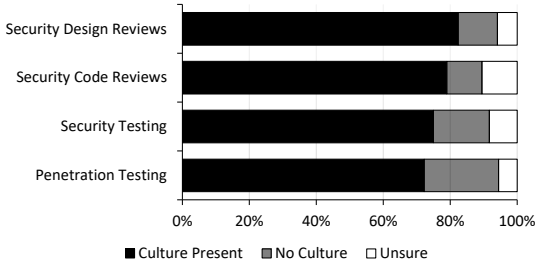
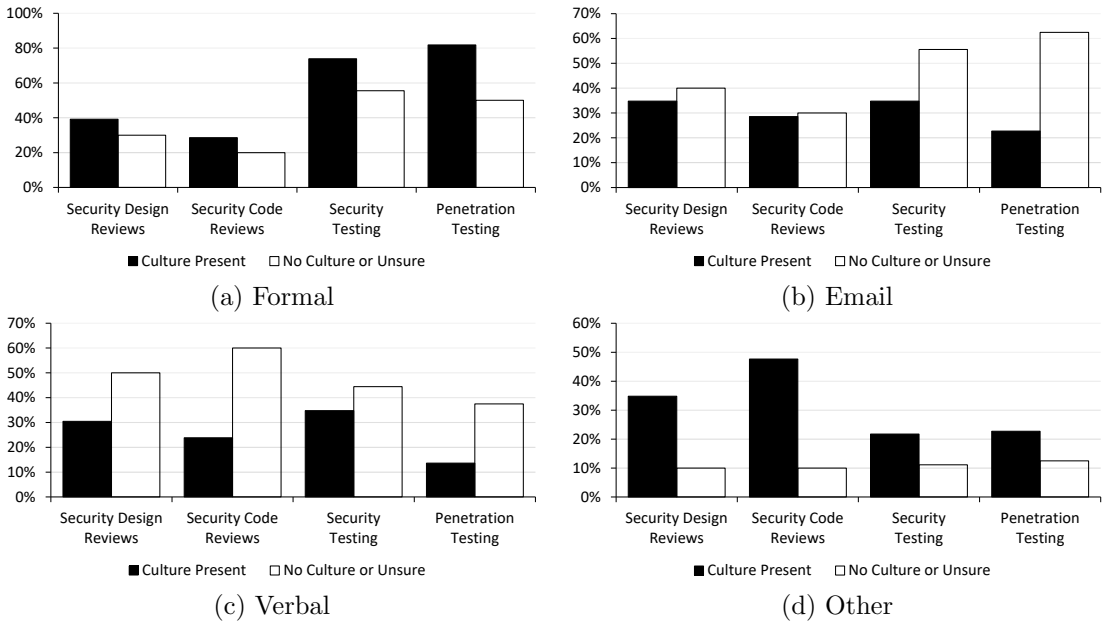


Figure 5.46: Influence of information security culture on governing processes

5.8.2.3 Existing Communication Forums

The influence an information security culture has on communication within the organisations is interesting. Specifically, the presence of such a culture seems to result in an increased use of formal communication between the various interactors, with a decreased use of email and verbal communication, see Figure 5.47.



(a) Formal (b) Email

(c) Verbal (d) Other

Figure 5.47: Influence of information security culture on type of communication

Notably, within the organisations possessing an information security culture, the primary interactors (namely, developers and test engineers) are the most informed when it comes to communicating the results of the security V&V activities. In addition, when an information security culture is present, there is more communication - between all types of interactor - for the test-based activities than there is within organisations without an information security culture.

5.8.2.4 Resource Flows

Figure 5.48 shows, without exception, that organisations with an information security culture invest more in security V&V than organisations without an information security culture (averaging 128% higher across all activities). Such investment is more likely to focus on the test-based activities, specifically, penetration testing.

Having observed that the use of tools and automation can be expensive (see Section 1.3.2), we aimed to understand whether the increased levels of investment within organisations with an information security culture are reflected in the levels of tool use

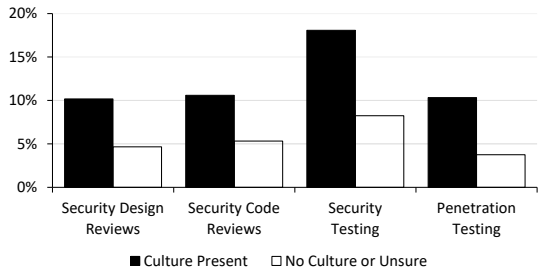


Figure 5.48: Influence of information security culture on investment levels

and automation being reported. Interestingly, in the instances where the use of tools are known to support an activity, and the presence or absence of an information security culture is also known, we find, as reflected within Figure 5.49, that the majority of cases where a tool is used an information security culture is also present.

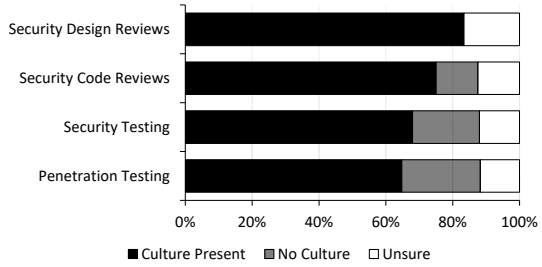


Figure 5.49: Influence of information security culture on tool use

We also find, see Figure 5.50, that the presence or absence of an information security culture corresponds with the amount of automation being reported. However, whilst the presence of an information security culture results in higher levels of automation being reported across the V&V activities in general - and lower levels of automation within organisations without an information security culture - we observe that the difference is much more pronounced in terms of the security V&V activities. As investment in automation is costly (see Section 1.3.2), these results provide indication that, within organisations possessing an information security culture, there is evidently an increased view that such investment is worthwhile.

Further, although we observed little investment in security V&V training (see Section 5.6.7), Figure 5.51 shows that such training is more likely to occur within organisations with an information security culture. Notably, the only organisations supporting security-focused review-based training are those with an information security culture. Similarly, the majority of the security-focused test-based training received ($\sim$72%) has occurred within organisations with such a culture. This, once more, emphasises that organisations with an information security culture are more likely to invest in the

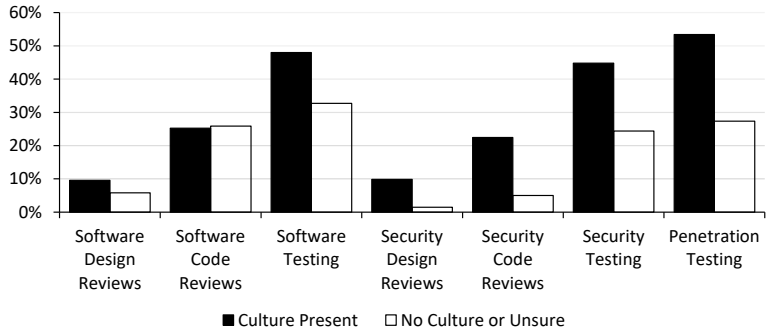


Figure 5.50: Influence of information security culture on automation levels

security-related activities residing within the software development lifecycle.

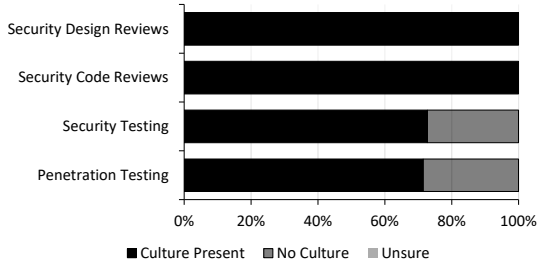


Figure 5.51: Influence of information security culture on training provision

The results, see Figure 5.52, also show that organisations with an information security culture believe more investment should be made in terms of test-based security V&V activities. Interestingly, these organisations do not feel so strongly about increasing the levels of investment in review-based security V&V activities. This possibly reflects an attitude, as expressed by some respondents, that they, for example: "*[f]eel that actively testing the software is more likely to find issues than code reviews*" [O:AV-I:JG]. In contrast, other respondents are keen to highlight that, for example: "*[g]etting the design right is the most important. The rest of the process will measure against that design*" [O:MC-I:AP]. However, as summarised by one QA Director [O:WS-I:WE]: "*I'm the head of QA for my organisation. For me testing is of the highest importance, as this validates that the other activities have been performed to an acceptable standard. If the other aspects such as Code reviews are not being performed correctly, the testing of the product will highlight these problems due to more failures in the application under test*".

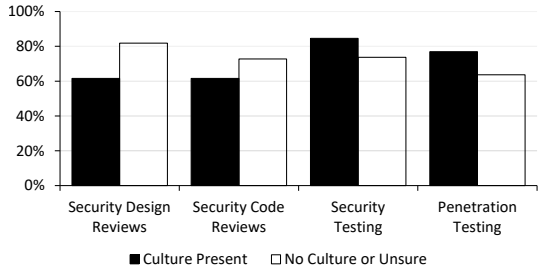


Figure 5.52: Influence of information security culture on whether more investment is required

5.8.2.5 Summary

Respondents, within organisations with an information security culture, are more likely to believe that the V&V activities being performed, are being performed well (see Figure 5.53). This holds for both software and security V&V, although it is more pronounced in terms of the latter. Notably, this belief is supported by increased levels of investment, tool use, automation and process governance.

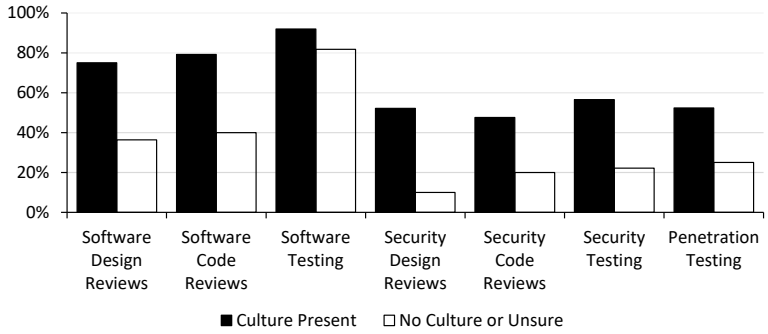


Figure 5.53: Influence of information security culture on whether the activities are perceived as being performed well

Interestingly, the presence of an information security culture does not dramatically change the importance of how the various V&V activities are viewed within the organisations. This was an unexpected finding, namely, that the afforded level of importance to the security V&V activities did not increase within the organisations reporting an information security culture. Specifically, the only noticeable change - when examining the results in their entirety - is that security design reviews are viewed with more importance than software code reviews. Supporting this we recall the view of one Test Manager [O:MC-I:AP], who succinctly stated: "*[g]etting the design right is the most important. The rest of the process will measure against that design*". Several respondents are also motivated in terms of cost, for example, a Head of Applications

Development [O:OS-I:ST] states that the "*[o]rder reflects the impact of defects and the costs to fix them. It is always cheapest to fix them in design and identifying them in code reviews is cheaper them testing as it is a stage earlier in the process*" (see Section 1.3.4). Similar is echoed by a Senior QA Manager [O:CR-I:MC].

Significantly, the results demonstrate that an awareness of security has permeated both organisations and - specifically, their software development lifecycles - thereby showing an information security culture can, and does, influence security V&V within software SMEs. In particular, organisations with an information security culture are much more likely to invest in the supporting tools and technologies necessary to perform V&V effectively. We find this in terms of resource flows, communication flows and the application of the activities themselves. Perhaps more importantly, we find that such organisations are more likely to perform the activities associated with security V&V and that an information security culture does result in an organisation with a more mature - and evidently appreciated - security V&V practice.

5.8.3 Reflections

Within this phase we have examined how security V&V is practiced, supported, and perceived, within UK-based software SMEs. This addressed the first research objective and provides an important contribution to knowledge by illuminating what was an empirical unknown [Kreeger and Harindranath, 2012, Kreeger and Harindranath, 2017]. We have also derived an understanding of the relationship which exists between such practice and an organisation's information security culture and the maturity level of an organisation's software V&V practice. This addresses the second and third research objectives respectively. Specifically, we found that a strong relationship exists between an organisation possessing an information security culture and the level of security V&V practiced, as well as the level of support afforded to the activities. We also found that the software V&V activities have typically reached a higher level of maturity. There was no strong evidence supporting the view that an organisation could possess a mature security V&V practice without an existing mature software V&V practice.

It was by focusing on both the interactors, and their various relationships, that a richer understanding of how the V&V activities were practiced, supported, and perceived, within software SMEs, was obtained. For example, subsequent to identifying both the human and the non-human interactors involved, it was by exploring the relationships within the network which helped generate a more nuanced understanding e.g. whereas several existing studies just enumerate the tools used (effectively, just performing interactor analysis), it was by performing the latter steps of STIN i.e. through network relationship analysis, that we were able to see how the tools were used by the

interactors, as well as what impacted their use. In contrast, if we had just performed some of the network relationship analysis steps we would have not been in a position to have fully appreciated and understood the results. For example, we found penetration testing the activity least impacted by a lack of time (which was, overall, the most frequently reported constraint across all the V&V activities practiced). However, it was from the interactor analysis performed that we were able to identify that a significant reliance on outsourced engineers existed for this activity i.e. any time constraints caused by the limited resources of SMEs would be removed by outsourcing the activity. These examples show how the various steps within STIN support one another.

In conclusion, the adoption of a survey proved invaluable in eliciting high-quality data concerning what is undoubtedly a sensitive subject (see Section 5.2). Further, the adoption of a socio-technical approach - namely, STIN - contributed, and ensured, that a more complete account of security V&V practice was attained. Within the next chapter, we continue to explore and address the research objectives - and the findings from this phase - by undertaking a second phase of data collection that prioritises the acquisition and analysis of qualitative data.

Chapter 6

Data Collection and Analysis: Phase Two

Within this chapter we present the second phase of data collection; whereupon, we analyse the data received and reflect upon the findings and the approach taken. Specifically, this chapter provides:

Section 6.1: The interview guide pre-testing performed.

Section 6.2: The adoption and application of thematic analysis.

Section 6.3: We analyse the data to build upon, and confirm, our findings from the first phase.

Section 6.4: We summarise the findings from the second phase and reflect upon the approach taken.

We begin by detailing the interview guide pre-testing performed.

6.1 Interview Guide Pre-Testing

With the second phase involving semi-structured interviews, this necessitated the development, and the pre-testing of, an interview guide. The importance of such pre-testing (similar to that of a survey instrument, see Section 5.1.1) has been readily acknowledged e.g. with it leading to guide refinement [Buchwald et al., 2014, Harrer and Wald, 2016], helping ensure quality [Slyngstad et al., 2008], validity [Novelli and Wenzel, 2013] and "the trustworthiness of the data collection" [Kangasniemi et al., 2015, p. 1041]. Notably, the interview guide was developed on the basis of the survey - which also underwent pre-testing - and through analysis of the data obtained during the first phase of data collection (Chapter 5).

Further, acknowledging the importance of selecting subjects from the intended population [Cooper and Schindler, 2014], we utilised subject matter experts when pre-testing our interview guide. As such, two separate pre-testing sessions were held - each with a subject matter expert - both of whom are software engineers, focused on V&V (averaging 6-10 years of experience), and who have worked for SMEs and large organisations. The interviewees involved with the pre-testing, and the sessions themselves (totalling approximately three hours), are detailed in Appendix J. The resulting interview guide is presented in Appendix K.

6.2 Thematic Analysis

Throughout the empirical software engineering literature, it is apparent that thematic analysis has been widely employed when analysing qualitative data [Wohlin and Aurum, 2015]. For example, when examining software engineer motivation, [França et al., 2014] apply thematic analysis to the data obtained from a series of semi-structured interviews. Similarly, [Lenberg et al., 2015] employ thematic analysis when studying the human factors which impact software engineering. It has also been utilised to understand the problems which can occur when users are involved in the software development process [Zowghi et al., 2015] and when investigating the challenges which surround continuous deployment within software organisations [Shahin et al., 2017]. As Wohlin and Aurum indicate, it is by adopting thematic analysis that it becomes possible to develop a deeper understanding of the data acquired [Wohlin and Aurum, 2015]. Therefore, and based on these examples - all of which show the successful use of thematic analysis, within an empirical software engineering research context, when studying data acquired from semi-structured interviews - we elect to employ thematic analysis during the second phase of data collection and analysis.

However, we observe that whilst qualitative data analysis, in general, has been considered as being both complex and mysterious [Thorne, 2000], thematic analysis itself has been viewed as being poorly defined [Braun and Clarke, 2006]. Given the limited guidance available to researchers on how to perform such an analysis [Nowell et al., 2017], we follow the six phases of thematic analysis as outlined in [Braun and Clarke, 2006]. Specifically:

- Familiarising self with data: as emphasised by [Braun and Clarke, 2006], it is necessary for a researcher to first become fully immersed in the data collected i.e. the data should be read multiple times, with any initial thoughts regarding analysis being documented.
- Generating initial codes: once an appropriate level of familiarisation with the data has been achieved, the coding process can begin i.e. the researcher can begin to identify patterns and interesting features within the data collected.
- Searching for themes: after the coding process has completed, the codes identified should be grouped together to form themes (with an individual theme representing patterns of meaning across the data obtained).
- Reviewing themes: the themes identified should be subjected to further review to determine whether they should be combined (potentially creating new themes), broken down further, or excluded from analysis altogether (e.g. if there is insufficient data to support the theme). As [Braun and Clarke, 2006] state, identified themes should be checked to ensure that a coherent pattern exists.
- Defining and naming themes: the themes identified should be named and defined appropriately, with consideration given to how the themes fit together to address the underlying research question.
- Producing a report: once the themes have been defined and named, it is necessary to construct a narrative around the identified themes.

Based on the phases above, we began thematic analysis by first familiarising ourselves with the data collected. Notably, as the author was involved in both the collection of the data - and its transcription - this helped ensure that a firm basis existed on which to further build familiarity with the data obtained. Therefore, whilst we previously acknowledged the time-consuming nature of interview transcription (see Section 4.4.1), we feel that the focus required to produce a verbatim account helped increase our level of immersion with the data. Further, it was during the transcription process, and

the iterative reading process, that we began to identify initial meanings and patterns within the data. As with others (e.g. [Gibbs, 2018]), we utilised a combination of paper and electronic-based approaches when performing thematic analysis i.e. we marked-up printed copies of the transcript (using a combination of pen and different coloured highlighters), as well as electronic forms of the data (with the mark-up occurring within a spreadsheet and the original transcript being stored as plain text). In summary, and aside from increasing our understanding of the interview data obtained, this phase resulted in some initial patterns within the data being identified; it also enabled us to capture any written notes taken during the interviews themselves (e.g. recording observed non-verbal gestures).

Once we had achieved a level of familiarity with the data, we began to expand on our notes and mark-up by identifying some initial codes. Effectively, we began to organise the data into meaningful groups that would ultimately lead to the development of themes. Given our adoption of STIN, in conjunction with a sequential explanatory mixed methods design (see Chapters 3 and 4) - with both the survey instrument and the interview guide being constructed around the themes within STIN (see Appendices D, H and K) - we acknowledge that this may have led to a purely theory-driven approach to coding. However, we elected not to be solely confined to the structure afforded by the adoption of STIN. Therefore, our coding approach was also driven by the data itself. We feel that adopting both a deductive and inductive approach to coding was important - especially when researching an empirical unknown - since, as acknowledged by [Seaman, 1999], semi-structured interviews can provide unexpected information, with all data having potential value (with [Braun and Clarke, 2006] also observing that the actual value of data may only become apparent later on in analysis). As captured by [Gibbs, 2018], these approaches are not mutually exclusive, with most researchers moving between them during analysis. Thus, and given our immersion with the data, we elected to adopt a manual approach to coding (continuing to use a combination of both paper and spreadsheet to achieve this).

Similarly, in terms of searching, reviewing and defining themes, we found that the identified codes started to coalesce under the main themes found within STIN, with each theme containing various sub-themes. For example, the overarching theme of "communication" (which maps directly to one of the steps within the STIN framework), contained a series of sub-themes: communication occurring internal to an organisation (e.g. the communication between the primary interactors and the influence Agile has on their communication), communication external to an organisation (e.g. communication with outsourced engineers and universities), as well as communication which crosses both an organisation's internal and external boundaries (e.g. communication concerning

defects and vulnerabilities). As stated above, the prescriptive nature of STIN helped structure the qualitative data into themes i.e. it helped prevent us "drowning in a sea of data" [Rowley, 2012, p. 263]. This was important since, as [King, 2004] cautions, it is possible to continually redefine definitions during the process of thematic analysis. However, once the themes were identified, we then started to build a coherent narrative around them, the result of which is presented within Section 6.3. As Braun and Clarke indicate, the analysis presented must provide sufficient evidence in order to justify the selected themes. Therefore, we, like others adopting STIN (e.g. [Meyer, 2007, Reinert, 2009]), structure and present our analysis based around the themes prescribed within STIN itself. However, whilst the overarching themes map to those within STIN, a number of the sub-themes identified went beyond STIN. For example, and given the inherent organisational bias of STIN [Meyer, 2006, Meyer, 2007], it was only by our also adopting a data-driven approach to analysis that it became apparent that an organisation's external environment has a clear impact on its security V&V practices (e.g. through the influence of customers and the reliance placed on outsourced engineers, as well as the communication with such interactors). Significantly, this resulted in the need to revisit the presented conceptual model (see Section 7.4.4).

In summary, by following the guidance within [Braun and Clarke, 2006], a more structured approach to thematic analysis was taken (e.g. ensuring that context was retained when coding extracts of data). It also led to our acknowledging that contradictions may be encountered, as well as that some extracts might fit into multiple groups i.e. be coded multiple times (as with others, e.g. [Pope et al., 2000], we also found that some coded data extracts could be included within multiple themes). Notably, the latter was encountered when adopting a theory-driven approach to the coding process (this is discussed further within Section 8.3.1).

6.3 V&V Practice

As with the first phase (Section 5.6), we utilise STIN to structure our analysis. Therefore, we begin by identifying the human and non-human interactors involved across the V&V activities (Sections 6.3.1, 6.3.2, 6.3.3 and 6.3.4), as well as their motivations, and incentives, for performing V&V (Section 6.3.5). We then examine the excluded interactors and the undesired interactions which exist within the V&V socio-technical network (Section 6.3.6). Interactor communication ecology is then discussed (Section 6.3.7) and the resource flows surrounding the activities are examined (Section 6.3.8).

6.3.1 Primary Human Interactors

The interviews confirmed that developers and test engineers are the primary interactors and that they exist as distinct interactor groups. This was found within all organisations, regardless of their adoption of Agile (where, we recall, the desired engineering generalist [Meszaros and Aston, 2007]). However, although distinct, both are situated physically (see Section 6.3.7), and hierarchically, within an organisation's engineering team (sometimes performing the same activity). As one Test Manager [O:RW-I:RL] indicated, although engineering "*sit as one big team*", they are "*split up into smaller scrum groups*" (each typically comprising 2-3 developers, 1 test engineer and a number of supporting engineers e.g. a business analyst and a configuration engineer). Similarly, a Head of Engineering [O:KD-I:MF] indicated their engineering team is "*split into four scrum teams, each working on separate products*". Whilst this structure is generally repeated across the organisations, some variances exist. For example, within some organisations (e.g. [O:CT] and [O:DF]) there is an appointed development manager and test manager who oversee the different engineering functions, both of whom report into a head of engineering (or equivalent e.g. a CTO). Within other organisations (e.g. [O:RV]) there are multiple development and test leads reporting directly into a head of engineering.

Hierarchically, the number of levels within an engineering team can also vary e.g. we found engineers, senior engineers, lead engineers, architects, managers and various combinations thereof, but all existing beneath one head of engineering (the underlying reporting structure between these levels can vary e.g. one Lead Software Test Engineer [O:RV-I:ML] looks after three inter-dependent teams). Finally, although the composition of a team can vary regarding developer-test engineer ratio, we found, within both small and medium-sized organisations (e.g. [O:RW], [O:DF] and [O:RV]), that this averages about 3 developers to 1 test engineer. We discuss the impact of team composition within Section 6.3.8.3.

There is also support for the findings concerning which activity a specific type of interactor was more likely to perform (see Section 5.6.1) i.e. we observed the negative view in which developers generally hold test-based activities (see Section 6.3.6) - confirming they are less likely to perform such activities - and that test engineers do not typically perform code reviews. This was evident in both small organisations e.g. "*they certainly do design reviews [...] but not the code reviews*" [O:SS-I:RD] (the Product Development Manager stated this was due to test engineers adopting a black-box approach) and medium-sized organisations e.g. "*it's just done within the developers themselves*" [O:RV-I:ML]. This view was supported by another Lead Software Test Engineer [O:RV-I:SK], within the same organisation, who noted there is "*not necessarily*

any test involvement in the code design stage". They attribute this to "*a different skill set*" existing between the two primary interactors and, although this has been observed by others e.g. [Dhaliwal et al., 2011], this clearly impacts communication (see Section 5.6.6). Therefore, the traditional developer and test engineer divide is still apparent, with one Software Tester [O:CT-I:DS] appropriately summarising this state as follows: "*the chances are we are going to carry on pretty much as we are with the devs doing all the code reviews of the source code and testers doing the functional testing*".

The interviews also confirmed that the primary interactors were more likely to perform the software V&V activities than the security V&V activities (thus reflecting the activity frequency observed within Section 5.5). For example, although the Software Tester referenced above discussed the code reviews and software testing being performed in some detail, there was evident uncertainty as to whether they would ever perform security testing e.g. "*if we get that far, security testing, or when we get that far, security testing*". They were also unsure whether security code reviews were performed, supporting the observed "Do Not Know" trend for the security V&V activities (see Figure 5.11).

However, whilst there was commonality across the organisations, in terms of the primary interactors, there was one notable exception identified through the interviews. Specifically, within one medium-sized organisation, we found a Security Test Engineer [O:RV-I:BM] (someone dedicated to performing security V&V within their organisation). Having worked for their organisation for over five years, they were first employed as a test engineer, transitioning into their current role ∼1.5 years ago. Interestingly, this was not their original intent: "*when I first left uni I wanted to be a developer, because I only knew about developers [...] my course was mostly Java, so it was like, okay, I want to be a junior Java developer. I went for a job and then I didn't get that, but then the recruiter said, oh, there's a tester role instead, and I was like, oh, I don't even know what that is*" (which emphasises the bias introduced at the curriculum level towards test-based activities, see Section 2.6). However, they have "*always had a big interest in security*" (the "*even outside of work, I'm doing security-related things*" attitude became readily apparent later in the interview when it was indicated: "*on the train this morning, I was playing around with new tools [...] somebody had recommended on Twitter*" - it was self-acknowledged that these extra-curricular activities "*feeds in*" to their work). It was this focus which resulted in their being perceived as a champion: "*I always kind of had a focus on security [...] it kind of became like people would come to me with security testing things*". Thus demonstrating the importance of a security V&V champion (see Section 6.3.6 for related discussion).

Further, whilst the organisation has several distinct teams (of varying levels of

maturity in terms of embracing Agile), the Security Test Engineer is not positioned within a specific team: "*I'm not really in any team but get involved in every team*". This supports the pervasive nature of security. However, whilst they report into one of the organisation's Lead Software Test Engineers, they also report into the Head of Engineering. Thus, although they consider themselves "*a bit of a lone wolf*", and neither a developer or test engineer ("*I'd like to think in between*" the two), they are active across design, code and test-related activities e.g. "*so sometimes if there's a big release, like an enterprise release that went out just before Christmas, I did help out with testing there just because there was less security stuff and I was free to do that*" and "*today I'm right now reviewing PHP code*" - therefore, although "Test Engineer" is in their title, they "*spend a lot of time talking to developers*" (see Section 6.3.6 for further discussion on the developer and test engineer relationship).

Interestingly, within the interviewee's previous company (a large organisation), a similar pattern was observed - namely, they were employed as a test engineer and then showed interest in security. However, within their current organisation, "*people got to kind of know that*" focus and "*then, just over the years, it just kind of gradually developed where it was more of a formal thing where things would have to be security reviewed by me at various stages*". The interviewee then approached the Head of Engineering: "*I said this is what I enjoy doing. I love security stuff. Test stuff is fine, but I think, you know, we've got a need for a security role full time. I'm virtually doing it full time anyway, can I have the job title as a security test engineer?*". Although a good example of an individual driving positive change, the Head of Engineering clearly felt the need to formalise the role (which recognises the importance of management support - we recall that a lack of management support was the second most reported issue affecting security V&V, see Section 5.6.5). Lastly, although this organisation is an exception in terms of formalising any aspect of security V&V into a focused role, they only employ one such individual. As the interview with the Security Test Engineer - and during the other interviews conducted within this organisation - progressed, it became apparent that this resource was heavily replied upon by others (we discuss this aspect further within Section 6.3.8).

6.3.2 Secondary Human Interactors

The interviews supported the observed reliance on outsourcing (see Section 5.6.2). Specifically, SMEs outsource their software and security V&V to a variety of organisations, based within several different countries e.g. Russia, Ukraine and India. The level of reliance can vary. For example, within some organisations outsourcing is considered "*a one-off activity*" [O:SS-I:RD] and "*not the norm*" [O:AG-I:DS], or it is a

regular activity, at a specific point in the development process e.g. with security V&V activities occurring "*[a]fter the development phase*" on a given project [O:VS-I:GH]. The interviews also confirmed the observed desire, by both small and medium-sized organisations, to further increase their use of outsourcing - which was particularly evident regarding the security V&V activities. For example, a Test Manager [O:RW-I:RL] vocalised: "*security, no, we don't outsource it, and I think that it is probably a very good idea to do that*" (this organisation currently only outsources their software V&V activities - highlighting it is "*mainly for, I suppose, like validation, pre-UAT things like that*"). An equivalent view was held by a Product Development Director: "*[a]n amount of black-box security testing could be outsourced, and perhaps should be outsourced*" [O:SP-I:WS] and a Senior Test Engineer [O:DF-I:JR] indicated their security V&V should be outsourced, since: "*there are experts that can find, analyse and help resolve items much quicker*". Similarly, a Software Tester [O:CT-I:DS] admitted: "*I certainly haven't got the [security] knowledge, [the Test Manager] sure hasn't got the knowledge*". Such recognition led the Senior Test Engineer's organisation to actively pursue the outsourcing of their security V&V (even though a previous attempt to outsource their software V&V reportedly "*left the company in a near state of ruin*"). Whilst this identifies a healthy degree of conscious incompetence [Thomson and von Solms, 2006], it emphasises the importance of improving the level of security V&V training within SMEs (see Section 5.6.7 for further training-related discussion). We continue to examine the impact of outsourcing on security V&V in Section 6.3.6.

In terms of customers, the interviews confirmed that any direct involvement - with either the software or security-focused activities - is infrequent at best (this is reflected within Figures 5.12 and 5.13). One Software Tester [O:CT-I:DS] vocalised the view that customers are welcome to test their products, but highlighted: "*it's the responsibility of [the company] to make sure our products are secure, it's certainly not their responsibility*". Whilst the interviews support the limited engagement of customers overall, they also confirmed their complete absence from any code-based review activities and showed, when involved, that they predominately focus on test-based activities. Generally, a customer's involvement is more indirect. For example, customers were more likely to request evidence showing that testing had been performed e.g. one Head of Engineering [O:KD-I:MF] states: "*[m]any of our customers insist on a penetration test report so that they can see what vulnerabilities we have and how we are addressing them*" (customers are known to request information on test maturity and ongoing test improvement activities [Hass, 2014]). This was echoed within other organisations (e.g. [O:RV]), where customers requested information on how an organisation had addressed, or was impacted by, specific vulnerabilities. Customers also showed interest in

product quality in general. For example, a Product Development Director [O:SP-I:WS] indicated their CTO was "*very much in touch with the major customers, for whom verification, validation and quality are constant subjects of discussion*". Therefore, whilst some customers are directly engaged with an organisation's V&V activities (see Section 5.6.2), they are, in general, more likely to influence them (see Section 6.3.5).

Whilst several references were made to Agile-related roles (such as product owner and scrum master), this was often in passing - even within organisations proclaiming to practise Agile (which is the majority in some form or another, see Section 5.4.1) e.g. within one small organisation [O:CT] the roles were by no means filled: "*at the moment the reality is that we haven't got a product owner, we kind of had a scrum master but he's kind of doing other stuff now*". We attribute this limited focus to the multiple hat syndrome found within SMEs [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018]. A view borne out by the scrum master, within the small organisation above, also "*doing other stuff*"; as well as when observing that one of the interviewees, based within a medium-sized organisation, is not only a practising test engineer day-to-day, but also holds an Agile-related position (namely, the Scrum Master and Lead QA Engineer [O:VS-I:GH]). This has also been identified by others e.g. within one software SME, the product owner, due to multiple hats, was "stretched pretty thin" [Block, 2011, p. 238].

Further, developers would typically assume the role of product owner - this was the case in both small and medium-sized organisations e.g. [O:SS] and [O:RW]. This also helps explain the limited engagement of product owners with V&V i.e. echoing the reluctance of developers to actively embrace test-based activities (see Section 6.3.6). Notably, the existing literature supports developers assuming the role of product owner e.g. [Jovanović et al., 2017]; with [Diebold et al., 2015] identifying the reason that a developer assumed the role, within one small organisation, as being due to organisational size. Additionally, and based on the author's own experience of performing the product owner role (albeit within a large organisation), the vast majority of individuals performing this role were promoted from within the development team - and it was readily apparent that some were more engaged in review and test-based activities than others. Specifically, those possessing a test engineering background were more thorough, and pro-active, when it came to the test-based activities; whereas, those with a development background, were often more active with the code review-based activities (thereby reinforcing, and supporting, the typical role divide observed between developers and test engineers, see Sections 5.6.1 and 6.3.1).

The interviews did not identify any additional secondary interactors.

6.3.3 Tertiary Human Interactors

The first phase identified two sets of interactors (primary and secondary, see Sections 5.6.1 and 5.6.2), both of which have some level of direct involvement with one or more V&V activity. However, the interviews identified a third set of interactors which, although not directly involved in performing, or owning, any activity, their indirect influence helps shape them, and impacts the interactors directly involved.

In terms of influencing activities, several examples show the tertiary interactors applying pressure to reduce the scheduled V&V effort in order to be first to market or to sell a product (echoing the findings of others e.g. [Nguyen et al., 2006]). We also find examples showing their impact on the primary interactors can be both positive and negative. For example, within one medium-sized organisation [O:VS], the customer service team appreciated the test team's help and input and, within one small organisation, a Software Tester [O:CT-I:DS] vocalised that the "*project manager guy [. . .] is doing a very good job actually of shielding us from all the pressure [from the management team]*". However, within another medium-sized organisation [O:RV], marketing were vocal in expressing their annoyance that engineering take too long to deliver new functionality. All of these examples - involving the tertiary interactors - obviously impact the interactors directly performing, or owning, the V&V activities e.g. we know that test engineers are motivated by praise [Hass, 2008] and that a lack of management support can result in a lack of motivation and interest [Perry, 2006].

Although we do not explore the tertiary interactors in detail, we find some commonality in the interactor groups existing within both small and medium-sized organisations. Specifically, the following interactor groups are well represented (for simplicity, we have reduced the naming of these groups e.g. "*[c]ustomer support division*" [O:RW], "*customer service*" [O:SS], "*support*" [O:KD] and "*[s]ervice desk*" [O:VS] are all represented as "customer support"): sales, marketing, customer support and management (where the latter includes CEOs, MDs and anyone on "the board"). Lesser interactor groups (i.e. those less frequently seen), include: configuration, project management and training. The impact of these groups, on the V&V activities, is discussed in Section 6.3.6.

6.3.4 Non-Human Interactors

Whilst the interviews did not identify any additional non-human interactor groups, they did confirm, and elaborate, several findings from the first phase regarding the tools and processes being used.

6.3.4.1 Supporting Tools

Aside from supporting the findings from the first phase - as well as identifying several different tools (such as the *Elevation of Privilege* card game to introduce people to threat modelling) - the interviews provided some additional detail. For example, within one small organisation [O:CT], the open source defect tracking tool *Bugzilla* was used until moving to the proprietary *Jira*. Similarly, the desire to use some "*proper bug tracking tools as well, rather than something that somebody had knocked up*" was the cause, according to a Lead Software Test Engineer [O:RV-I:SK] within a medium-sized organisation, to move from an in-house developed tool to *Jira*. Whilst both examples support the overall prevalence of *Jira* (its popularity is echoed by others e.g. [Würsch et al., 2013]), it should not counteract the finding that there is, generally, more use of open source tools than proprietary ones. For example, although the small organisation referenced above has moved to *Jira*, all other tools discussed by the Software Tester [O:CT-I:DS] are open source, therefore, continuing to support the finding that smaller organisations are more likely to use free or open source software [Andersson and Runeson, 2002, Sitnikova et al., 2007]. It also shows a tendency to adopt "popular" tools - which is evident, in the context of *Jira*, both within organisations adopting Agile [Goth, 2009] and within SMEs [Mishra and Mishra, 2013]. Additionally, multiple references were made to *pytest* as the adopted Python testing framework, further echoing the use of popular, open source tools [Smelter and Moseley, 2018].

However, the interviews also showed that a combination of open source (e.g. *TestLink* for test management and *Git* as a version control system), proprietary (e.g. the vulnerability scanner *Acunetix* and the static code analysis tool *PVS-Studio*), and tools developed in-house, are used across the V&V activities (thus reflecting the findings from the first phase, see Figure 5.17). They also showed that a variety of communication tools (e.g. *Skype* and *Slack*) and mechanisms (email and telephone) are used by the interactors (see Section 5.6.6).

6.3.4.2 Governing Processes

We found further support that the software V&V activities are more likely to have a governing process. For example, the responses when posing: "Does your organisation follow any secure development process?", varied from "*I don't believe so*" [O:AG-I:DS] and "*[n]o, not that I am aware of*" [O:RW-I:RL], to an outright "*no*" e.g. [O:DF-I:JR] and [O:CT-I:DS]. This certainly helps support the findings from the first phase (see Section 5.6.3.2).

The interviews also afforded a more detailed view on the scope of the processes

reportedly in place. In particular, one Product Development Director [O:SP-I:WS] was clearly able to elaborate on what constituted their organisation's development process indicating, for example: "*[c]heckin to the code base is not allowed unless certain V&V has passed*" and "*[n]ightly build processes run further tests to ensure there are no further reaching effects of the latest checkins*". However, in contrast, they simply stated that "*[s]ecurity V&V is currently a more pre-release test*". Similarly, a Scrum Master and Lead QA Engineer [O:VS-I:GH] indicates that whilst software V&V is applied during the development process, security V&V occurs "*[a]fter the development phase*" (this is predominately due to such activities being outsourced) and is primarily shaped by "*lessons learned*". These examples further show that security V&V, from a governance and process perspective - and thus regarding practice as well - sit at a lower level of maturity. It also reduces the opportunity of using a process as a means of motivation (see Section 6.3.5.2).

Further, within some organisations, a single development process covered both development and test-based activities (thus covering activities performed by developers and test engineers). Within other organisations there were instances of specific processes existing e.g. one Product Development Manager [O:SS-I:RD] discussed "*ensur[ing] that the testing's done in accordance with the company's test policy*". However, when seeking elaboration on whether an organisation's existing processes covered the security V&V activities practiced, we found they were variable in scope e.g. a QA Test Lead [O:CD-I:KS] indicated a set of "*basic coding guidelines*", to protect against "*some generic threats*", comprised their secure development process.

There was also variety, where a secure development process existed, in terms of its origins, as well as what influenced its adoption and development. For example, within one medium-sized organisation [O:SP] the secure development process was developed in-house and did not draw inspiration from any established secure development process i.e. the Product Development Director [O:SP-I:WS] stated that the process was "*not a formally recognised one*". The Head of Engineering [O:KD-I:MF], at another medium-sized organisation, indicated that whilst their secure development process was also not based on any existing process, its development was driven by the organisation achieving ISO27001 certification (resulting in their being "*required to have development practices that demonstrate security and risk are considered*"). Notably, we found only one instance of an organisation employing a formally recognised secure development process i.e. the use of Microsoft's SDL [Microsoft, 2020] within a medium-sized organisation [O:RV]. Interestingly, this adoption grew from a customer support ticket asking: "*what kind of security process do you do*". This resulted in the organisation's Security Test Engineer [O:RV-I:BM] investigating: "*I looked into it and I was like, oh, actually, we do*

the Microsoft one, that's what we do, and so then it was like, actually, we don't do that bit, let's do that as well" (this example also shows the potential impact of customers, see Sections 6.3.2 and 6.3.5, and conveys the importance of having security-focused resource within an organisation, see Section 6.3.1).

6.3.5 Incentives

The first phase identified several factors motivating the interactors into performing V&V (see Section 5.6.4). The second phase allowed us to explore the four primary factors in greater depth.

6.3.5.1 Improving Software Quality

Within Section 5.6.4.1, we found, overwhelmingly, that people are primarily motivated to perform the various activities through a desire to improve software quality, and through identification with the work. Exploring this during the interviews, supports the view that interactors are often motivated by intrinsic means. Specifically, within several organisations, interactors were not directly incentivised for achieving a secure product - as stated by one Scrum Master and Lead QA Engineer [O:VS-I:GH]: "*no incentives are offered - it's part of the job*". Whilst this might imply their organisation sees little value in offering incentives to perform security V&V, on exploration, it appears their test engineers are motivated by the "*simple reason [of] personal pride in their work*". Thereby echoing identification with the task [Hall et al., 2008], an attitude reflected by the Test Manager [O:RW-I:RL] within another organisation: "*[t]he way we like to look at it is whatever I do in my job has a direct impact on someone else no matter how big or small, positive or negative*". This is a positive finding, further supporting the view that interactors are genuinely motivated by intrinsic factors.

Further emphasising affinity with a task, one QA Test Lead [O:CD-I:KS] stated: "*our objective is always to release a bug free product*". Exploring the extent people were prepared to go to achieve this, we found that if they were faced with a "*critical deadline*", and additional effort was required (i.e. "*extended working hours*" or "*working on weekends*"), the incentives offered by their organisation were minimal. Specifically, their medium-sized organisation was willing to "*take care of [the] generic necessity*" (which was described as covering food, transport and accommodation). Although minimal (i.e. there was no mention of overtime, bonuses or time off in lieu), this was sufficient to motivate people in meeting their objective of releasing a product without defects. This attitude was prevalent throughout the organisation - regardless of interactor type - as it was highlighted, and reinforced, that development "*is a*

collaborative team effort to make a successful product".

In some cases, people were motivated to improve software to avoid the "*serious consequences*" resulting from any security failings (such as those captured within Section 1.3.4). For example, within one organisation, who develops products intended for a particular medical context, there were two factors motivating them to develop secure products: whilst their "*primary mission was patient safety*", the result of "*any data theft could have had serious consequences in terms of both patient safety and the public relations of the company*" [O:AG-I:DS]. Thus showing regard for the consumer of the product, the context in which it will be deployed, and the consequences of failing to develop a secure product to the organisation itself. Such awareness was also reflected within another organisation e.g. "*[w]e are all very aware of the affect on the company, in terms of from a PR basis, on what security risks can do to a company*" [O:RV-I:SK].

6.3.5.2 Mandated by Process

Within many of the organisations (both small and medium-sized) we found an absence of any form of formal secure development process. However, this was not as black and white as having a governing process or not. For example, within one medium-sized organisation, whilst there was no overarching secure development process, there were "*some basic coding guidelines [...] which make the product secure from some generic threats*" [O:CD-I:KS]. Similarly, within another medium-sized organisation, although no secure development process is used, they rely "*mostly on regressions from lessons learned*" [O:VS-I:GH]. Whilst utilising such knowledge is commendable (and is supported by security maturity models e.g. [McGraw et al., 2016]), this reliance suggests that the organisation has always been able to recover from the incident which prompted the lesson. Acknowledging the potential impact of security vulnerabilities (see Section 1.3.4), this will not always be achievable. However, even this was an improvement over one small organisation [O:CT], where no form of secure development process was followed.

Further, within one medium-sized organisation, whilst a secure development process was reportedly in existence, the Product Development Director [O:SP-I:WS] clarified: "*although not a formally recognised one*". The general absence of reference to a formally recognised secure development process suggests: that SMEs are either unaware of their existence, or they are not suitable for such a context. However, more often than not, there was not a strong understanding of what was meant by having a secure development process, or whether one existed. For example, on posing the question does your organisation follow any secure development process to one Test Manager [O:RW-I:RL], clarification was required: "*[w]hat do you mean? I am not totally sure*

what you mean by secure development process?". Explaining, and providing examples, the response was "*[n]o, not that I am aware of. I just wanted to make sure I knew*". Similarly, a Lead Integration Engineer [O:AG-I:DS] indicated "*I don't believe so*". This example is worrying for several reasons, not least that the organisation is small, but that this is an engineer of some seniority - with established tenure - within an organisation where the importance of security was both highlighted, and acknowledged, due to the sensitive environment in which their products are deployed. Notably, only a single organisation employed a formally recognised secure development process. This was within a medium-sized organisation [O:RV], however, of the three people interviewed within this organisation, only one was able to vocalise this. This supports the view that there is a general lack of awareness within organisations as to whether a secure development process exists.

6.3.5.3 Instructed by Management

Although management is an important factor in determining the success of a software project [Peters, 2003], we found that the identified interactors are generally not motivated by management when performing software or security V&V (see Section 5.6.4.2). This was also reflected during the interviews. For example, although the Head of Engineering, within one medium-sized organisation [O:CD], was considered as being "*always above on all*" and the "*decision maker*", they did not influence any part of the software development lifecycle, thereby reflecting they were more of a figurehead, generally removed from the day-to-day engineering activities (the author has observed this within larger organisations, but less so within smaller organisations). Whilst this suggests, as an organisation grows, there is an increased distance between management and engineers, we find, when it comes to the various security V&V activities being performed within one small organisation [O:AG], a tone of evident frustration - namely, that their VP of Engineering "*ought to have had plenty of input, but he sat back and let the development manager take most of the responsibility*". This emphasises management influence was minimal within this organisation when it came to security V&V.

However, within one small organisation [O:SS], the development and QA functions each have "*a team lead or a technical lead*" who "*are responsible for ensuring standards in their own area*". Notably, their Development Lead "*will ensure that their team members perform the unit tests as part of the development process before that's then compiled into a build and then passed over to the QA team*". This is an important function - since not only has developer motivation for performing unit testing been shown as lacking e.g. [Runeson, 2006], the attitude held by the developers within this

organisation - with regards to all test-related activities - was succinctly summarised by the Product Development Manager [O:SS-I:RD] as: "*they prefer not to have to do it*". Therefore, it was only by having a senior member of the development team, with the necessary authority, that such an attitude could be overridden - thereby leading to unit testing being regularly performed. However, it is important to highlight the difference existing between this senior member of the development team and the Head of Engineering who was considered more of a figurehead: namely, the Development Lead is a technical individual actively participating in the day-to-day engineering activities e.g. by undertaking code reviews and performing peer reviews. This has probably helped engender a level of trust and respect by other members of the development team. Similarly, it is the QA Lead "*whose responsibility it is to then ensure that the testing's done in accordance with the company's test policy, and ensuring that the software is fit for purpose*".

Although we acknowledged the absence of Agile-related roles (see Section 5.6.2), within one organisation [O:RW] we found a product owner acting as the interface between the engineers and management: "*promising things to the board and then we have got the coal face*". The divide between these interactor groups is evidenced by the following colourful comparison: "*I guess it's like the military: one does not understand strategy and one does not understand how to fight*" [O:RW-I:RL]. However, although there is evident distance between management and those performing V&V, within one medium-sized organisation [O:RV], the Head of Engineering is as involved in security as anyone, since "*he bears the brunt of it the most if we had a very public security issue*". Further, it is apparent that this Head of Engineering "*keeps across all vulnerabilities*", communicating and working closely with the Security Test Engineer within the organisation and, although stated somewhat in jest, the Head of Engineering's involvement might just be because "*the one does not want to be caught out by the other*". Notably, this level of involvement and engagement by management - across the activities - appears to be an exception. It also further conveys the importance of management (and people in roles of seniority) being both engaged, and technically astute, in order to successfully motivate people. However, this alone does not guarantee their ability to encourage members of their team since, within one small organisation, a Senior Test Engineer [O:DF-I:JR] indicates: "*[t]he head of engineering is from a development background, although doesn't have a massive influence over the level of software V&V*" (which might be attributable to the relationship between developers and test engineers, see Section 6.3.6).

6.3.5.4 Requested by Customers and External Factors

Whilst software security has been considered a 'market for lemons' [Anderson, 2001], one Head of Engineering [O:KD-I:MF] stated that: "*[m]any of our customers insist on a penetration test report so that they can see what vulnerabilities we have and how we are addressing them*". Further, some customers proactively ask organisations about well-known vulnerabilities to determine whether their products are impacted. For example, a Lead Software Test Engineer [O:RV-I:SK] emphasised: "*[t]he nature of our software ... soon as that gets equated with a security risk, that makes a lot of people quite nervous*" (Heartbleed [Codenomicon, 2014] was a specific example given of where customers enquired). Obviously, not all customers may have such requirements, or are even conscious of having such requirements. However, such a group is likely to diminish with the cost of security failures becoming increasingly apparent, as well as through the demands of updated regulation (both through certification and legislation).

Such regulation makes a variety of demands on software producing organisations. For example, one medium-sized organisation [O:RV] is currently working towards three standards. As captured by one Lead Software Test Engineer [O:RV-I:ML]: "*we've got to use certain crypto libraries here and there*", therefore, "*it's going to need a lot more work to make sure we're compliant because we need to use specific libraries*". In another example, the Head of Engineering [O:KD-I:MF], at another medium-sized organisation, indicated: "*[w]e are also ISO27001 certified, which requires us to address security vulnerabilities*" and elaborated that: "*[a]s part of our ISO27001 certification, we are required to have development practices that demonstrate security and risk are considered*". However, achieving such certification often has at its basis the ability to enter new markets - as summarised by one Lead Software Test Engineer [O:RV-I:SK]: "*[f]rom a sales point of view, you can't make sales to certain parts, to certain industries, if you aren't certified*". The Security Test Engineer [O:RV-I:BM] echoed similar: "*the driver is to get into new markets there*". On inquiring where this driver originated, it transpired that the Security Test Engineer and the Head of Engineering "*started looking into this late last year or the middle of last year*", although it was the sales team who "*bumped up the priority of this, because they said, you know, we've had people approach us asking about these kind of things*". Therefore, achieving such certification - which impacts the security properties of the product and the associated V&V - is primarily driven by interactors outside of engineering - and not the interactors directly involved with V&V. This was supported by further discussion, where it became apparent that the organisation's employed defect tracking system (*Jira*) contained stories "*that are years old and saying, you know, we should do this*". The Security Test Engineer later acknowledged: "*[s]o, I guess the driver is from the sales team, the sale guys, them*

saying, you know, we had people approach us about this, and therefore it got bumped up, so its them driving it".

Within one small organisation [O:DF], whilst performing V&V is motivated by "*the clients we sell to*", this also influences the nature of the V&V being performed. This is due to changes in this organisation's targetted software market, where "*[t]raditionally the product would sit on enclosed networks, where external exposure was extremely limited*". However, as stated by the Senior Test Engineer [O:DF-I:JR]: "*over the last few years our users within the market place have started to change and how the product is accessed also*". This necessitated a change not only in V&V focus, but product security in general, whereupon, the organisation's Head of Engineering "*is starting to look more into the security side of parts of the product*". As noted above, the shape of a product, and the subsequent V&V being applied, is also influenced by external certification requests - which are not only driven by an organisation wishing to enter new markets, but also by the organisation's customers.

There was also acknowledgment of competitors. For example, within one small organisation [O:DF], being "*more secure than our competitors*", translated to the incentive of achieving "*better sales*". Further, within one medium-sized organisation [O:RV], it became evident competitors were kept in mind: "*[w]e have a competitor that had something of a security breach not so long ago and it didn't affect us because of the way our software works, but we've changed the way our software works since Christmas. So, the problem facing this competitor is something that we have to be aware of as well*". That software SMEs operate in competitive markets [Feldmann and Pizka, 2003], it is perhaps unsurprising to see some acknowledgment of competitors. However, the number of direct references made to competitors acting as a source of motivation was limited.

6.3.5.5 Conclusions

The interviews confirmed that improving software quality is the primary factor motivating interactors to perform V&V; thus confirming the results of the first phase (see Section 5.8.1) and other studies examining motivation in software engineering e.g. [Beecham et al., 2008, Hall et al., 2008]. However, whilst the basis of this motivation appears to be predominately intrinsically driven, we cannot discount that other supporting factors may be at play. For example, within one medium-sized organisation, where "*[t]he teams as a whole are motivated to produce good quality software - it is the ethos of the development and validation departments*", it became apparent, on further discussion, that such motivation may not solely be driven by a desire to achieve a quality product - as stated by the Product Development Director [O:SP-I:WS]: "*[w]hen a*

successful release is made then there are sometimes bonuses paid to those who have been heavily involved in that particular release". However, this was the only example found of a financial incentive encouraging interactors to improve software quality.

Aside from highlighting a lack of awareness, as to whether a process exists and covers the security V&V activities practiced, the interviews also showed a lack of understanding regarding secure development processes more generally. Notably, only a single instance was found where a recognised secure development process existed and was actively followed, however, even this apparent exemplar was not without blemish - since whilst the organisation's Security Test Engineer [O:RV-I:BM] helped instigate its adoption, we found this was initially driven by a customer. On asking the interviewee whether people within the organisation were aware of this process, the start was positive: "*[s]upport people would know that because they relay that to people externally*". However, things then became less certain: "*I would hope the engineering team would know that. HR, would they know that, probably not. Would the sales guys know that, maybe not, maybe not*". Of these, engineering should be aware of the process. Unfortunately, having interviewed two other members of the engineering team within this organisation, neither appeared aware. This emphasises the need to improve awareness so as to establish consistency in practice.

We also found confirmation that management is not a strong motivator (see Section 5.6.4.2). However, we find a more nuanced view developing, namely, management should be seen as being actively involved in the V&V activities. In the instances where there were signs of such engagement, interviewees spoke much more positively about their influence on V&V (both software and security-focused). Also supporting the results from the first phase, there is little reference to being motivated by a sense of fun - aside from a single reference to the *Elevation of Privilege* card game which, although now not used as often, was: "*kind of just, you know, gamification, to get people interested and, you know, excited, you know, what's this?*" [O:RV-I:BM]. Similarly, in terms of confirming the results from the survey, we found no signs of performing an activity to simply tick a box.

However, customers are clearly a strong motivator, although this appears dependent on the industry sector and the nature of the product. For example, a Product Development Manager [O:SS-I:RD] was quite emphatic when stating that "*ensuring a quality and secure product is one of the key requirements in what's required by our customers*", explicitly vocalising that meeting these requirements is of "*paramount importance especially in the area in which we work, which is healthcare*", with a failure to deliver upon these impacting the company's reputation. Likewise, a Lead Integration Engineer [O:AG-I:DS], whose organisation also develops products for the healthcare

sector, indicated they are motivated to develop secure products to avoid: "*serious consequences in terms of both patient safety and the public relations of the company*".

Finally, whilst we examined the identified factors of motivation in isolation, it is important to acknowledge that they can overlap. For example, crossing the boundary between motivation by process and instruction by management, a Product Development Manager [O:SS-I:RD], within one small organisation, indicated that the engineering team (encompassing developers and test engineers) are motivated to perform security V&V since "*it's part of their job description*". Further, within one medium-sized organisation [O:SP], the CTO actively requires a focus on V&V because he, in turn, is "*driven from his day to day interactions with customers and their requirements*". Continuing with this theme, we find "*he is very much in touch with the major customers, for whom verification, validation and quality are constant subjects of discussion. It is very much the case that the need for such is fed down from above*". Thereby encompassing customer requests and instruction by management.

6.3.6 Excluded Actors and Undesired Interactions

Within the first phase, we found that several undesired interactions - both technical and social - impacted the V&V activities (see Section 5.6.5). The interviews enabled us to explore these in more detail. We begin by revisiting the excluded actors.

6.3.6.1 Actors Excluded from Security V&V

The interviews did not identify any additional types of excluded actor above those identified previously (see Section 5.6.5). Generally, we found the view that "*pretty much the ones that need to be engaged with it [security V&V] are*" [O:RW-I:RL], with several interviewees indicating a variety of interactors, throughout their organisations, are involved e.g. "*items are passed up through the chain of command as required*" [O:DF-I:JR] and "*we try to involve everyone, from top down, in terms of having a view, so there is kind of buy-in across the organisation of the requirement for producing secure software*" [O:SS-I:RD]. However, exploring some of the referenced interactors, we often find their involvement is limited e.g. within one medium-sized organisation the CTO is "*engaged*" with their security V&V activities, although, by being "*on the board*", they have "*a lot of responsibility so they are [engaged] as much as they can be*" [O:RW-I:RL]. Notably, whilst there appears an intent to involve everyone, one interviewee succinctly vocalised the general situation: "*security is the job of everybody, not necessarily everybody knows enough about it to contribute*" [O:CT-I:DS]. Otherwise, the interviews confirm the results from the first phase i.e. although the primary interactors should

be involved, universally, they are not (for example, within one medium-sized organisation [O:VS] it is reported that developers should be engaged); and that undesired interactions and resource flows inhibit interactor involvement with the various security V&V activities e.g. "*in an ideal world, a software tester with knowledge and experience of security testing would have been involved, but no such person existed within the organisation*" [O:AG-I:DS].

We now explore the identified undesired interactions.

6.3.6.2 Developer Perception of Test-related Activities

On posing the question: are developers happy to perform test-related activities, the response by one Test Manager [O:RW-I:RL] was immediate: "*[n]o, they are not*". Exploring this, the situation is not as black and white as it first seemed since, although developers "*wouldn't pickup up something and just test it*", we find "*they would test their own work*". However, typically, we find test-based activities are considered second-class activities [Bertolino, 2003] by developers. This is best illustrated by a Product Development Manager [O:SS-I:RD], who captured the view held by their organisation's developers as: "*[i]t's one of those, kind of, washing the dishes type jobs, you've got to do it but nobody likes actually doing it*". Notably, equating testing to a rather thankless, generally unpleasant task (developers have previously regarded software testing negatively e.g. [Rooksby et al., 2009]).

However, within some organisations, there is acknowledgment that even though testing might be considered an unpleasant task, it is one which needs to be done. For example, within one organisation, although developers perform test-related activities regularly, this appears somewhat grudgingly - as indicated by the Head of Engineering [O:KD-I:MF]: "*[w]hether they are happy to do this is a different matter*". This further supports the view that interactors are not motivated by a sense of fun (see Section 5.6.4.3). However, whilst the Head of Engineering acknowledges that "*[d]evelopers will prefer not to do those activities*" - reinforcing the second-class perspective - they "*recognise that they must be done so will pick them up*". This shows an identification with the task - and motivation to create a quality product (thereby echoing [Beecham et al., 2008, Hall et al., 2008]) - and supports the view that software engineers are motivated by conflicting needs [Fairley, 2009]. Exploring the reasons for their not wanting to perform test-related activities, the Head of Engineering stated: "*they would prefer not to create the test scripts or test scenarios*", but "*would rather pick up existing test scripts and run through them manually or write automation tests for them*". This suggests, within this organisation - where Agile has been adopted - that developers wish to avoid engaging too closely with test-related tasks i.e. by investing the time to design

tests. Specifically, there is a preference to execute or automate existing tests. Such an approach completely separates the developers from the creative aspect of test design, reducing the impact of their knowledge in helping shape the test approach. Given that many developers have not been exposed to software test design techniques [Copeland, 2003], and that manual test case design is time-consuming and requires considerable expertise and experience [Chawla et al., 2016], this may explain their reluctance in helping formulate test designs. Additionally, focusing on automation (although beneficial, see Section 2.4) appears to reflect a desire by developers to keep within their comfort zone of writing code. Notably, some developers prefer Agile environments, since: "they feel they can just write code and let other people worry about the details, results, testing, etc" [Crowder and Friess, 2015, p. 22], which supports the observed focus on test automation by developers, to the exclusion of test design.

Similarly, a Lead Integration Engineer [O:AG-I:DS] indicates that developers were never involved in software testing. This emphasises the traditional role divide between developers and test engineers even within organisations practising Agile. However, as a good working relationship existed between the two interactors (e.g. "*[d]evelopers and testers sat opposite one another and worked together closely*"), the interviewee indicated: "*so I suspect that they would have got involved if absolutely necessary*", although this was later qualified with: "*I can't say for sure whether they would have been happy to get involved*". Other respondents were less positive, with one Scrum Master and Lead QA Engineer [O:VS-I:GH] reporting that the second-class citizens [Juristo et al., 2006] attitude is still present within their organisation. For example, when asking whether developers are happy to perform test-related activities there was a firm response of: "*[n]o! They see it as beneath them*". Further, within some organisations, this appears to vary by developer e.g. "*some are happy to perform testing duties and are very vigorous*", however, "*some say they have [performed some testing], but receiving a new build it is quite obvious that they haven't*" [O:DF-I:JR]. Whilst the former attitude is pleasing, the latter clearly demonstrates a strong antipathy towards testing, running counter to the otherwise prevalent attitude of wanting to improve software quality (which, as we found, was the primary motivator for performing any V&V activity, see Section 5.6.4.1).

In summary, our findings support the view that test-based activities are generally considered unpalatable by developers (this was apparent when interviewing both developers and test engineers). This is true for both software and security-focused testing. However, it is particularly concerning when, given the choice, developers appear more likely to prefer to undertake software testing (e.g. "*because they know it inside out*" [O:VS-I:GH]), than security-focused testing (as "*it is quite a complex area*"

[O:RW-I:RL]; with a Product Development Manager [O:SS-I:RD] also emphasising that developer testing "*would not be of that complexity*"). Significantly, developers appear to require some form of encouragement to perform any form of test-based activity (i.e. with it becoming a formally recognised responsibility) e.g. with a Product Development Director [O:SP-I:WS] indicating: "*[a]ll developers do an amount of testing as part of the development process and are responsible for putting together auto tests for use once their software is in the final product. They also act as tester during the final system test of the product*". This emphasises the importance of actively motivating developers in performing test-based activities (for example, through ensuring and enforcing a process [Testa, 2009]).

6.3.6.3 Relationship Between the Primary V&V Interactors

Within the first phase, even with the wide adoption of Agile (advocating the engineering generalist [Meszaros and Aston, 2007]), all organisations distinguished between developers and test engineers (see Section 5.6.1). The interviews did not contradict this, with one Product Development Manager [O:SS-I:RD] - even though their organisation proclaims to practice Agile - clearly vocalising: "*there is a distinct separation*" between them.

In several instances, this strict role separation has negatively impacted the V&V activities due to developer and test engineer conflict e.g. a Scrum Master and Lead QA Engineer [O:VS-I:GH] states: "*[t]here seemed to be a them and us*" attitude prevalent throughout their organisation. This was further exacerbated by the developers not wanting to engage with the test engineers as "*they did not seem to like being challenged*". Exploring the issue of being "challenged" led to the understanding of: "*[c]hallenged in how the system worked mainly, if new functionality was being introduced they didn't want testers involved in discussions. It was like they felt we couldn't bring much to the table*". This is suggestive of a vicious cycle. For example, [Zhang et al., 2014] found that some developers reported conflict between themselves and test engineers due to the latter not possessing sufficient product knowledge. However, we found, within a well-established, medium-sized organisation, that such conflict has apparently resulted in the complete degradation of communication between the two types of interactor, whereupon, developers did not wish to engage with test engineers on how the product was evolving. This demonstrates a dismissive attitude entirely reminiscent of the reportedly held developer view of test engineers as second-class citizens [Juristo et al., 2006]. It also supports the adversarial nature of software testing, the conflict which can result between developers and test engineers [Sawyer, 2001, Cohen et al., 2004, Zhang et al., 2014] - and its impact on V&V (i.e. forcing test engineers to ex-

plore a black-box on their own, an approach not conducive to effective security testing [Thompson et al., 2002, Potter and McGraw, 2004]) - demonstrating that such attitudes have become firmly established within this organisation. Significantly, discord between the primary interactors can impact software quality [Zhang et al., 2018]. As we found (see Section 5.6.6), developers and test engineers need to communicate to perform V&V effectively. Unfortunately, a direct consequence of this attitude was that "*[i]t made the junior testers less likely to go to them [the developers] if there was an issue*".

Whilst the above was the most extreme example found, often the situation was less entrenched and vociferous e.g. one QA Test Lead [O:CD-I:KS] regarded the relationship between developers and test engineers as being "*some kind of love/hate relationship*". Specifically, they acknowledged that whilst test engineers need to find defects - and that developers "*treat this activity professionally*" - "*some are very particular to their work so they take bugs as ego*". This results in some developers becoming frustrated. However, and given the closeness between the primary interactors within this organisation, this frustration does not necessarily have a negative impact. As the QA Test Lead summarises: "*at the end of the day we are a team, so it's nothing personal to anyone*" - importantly, adding: "*and it works like this*". Thus any frustration is momentary (which contrasts to the example above, where such attitudes have pervaded the engineering team and become normalised).

Seniority also played a role in the relationship between developers and test engineers. Notably, within the organisation where junior test engineers were less likely to communicate directly with developers (because of the observed level of conflict within the organisation), it became apparent, when exploring whether this conflict was a result of age or tenure, it "*[s]eemed to be the case with the guys that had been with the company a while, newer ones to join were keener to help for a while until they learned to work like the others*" [O:VS-I:GH]. This emphasises that a negative culture had developed and was shaping the view of other developers towards test engineers, echoing [Shah and Harrold, 2010] in that the attitude of senior members can influence the attitude of others towards testing. However, we find this appears to be less concerned with a specific activity, but the relationship between two types of interactor, thereby displaying a greater detrimental impact socially.

Exploring the concept of seniority, and its impact on the interaction between developers and test engineers with other interviewees, it became apparent that such undesired interactions were more the result of individuals rather than any ingrained organisational culture that could be generalised across either type of interactor. For example, although technical expertise and age were touched upon when discussing the impact of

seniority, a Security Test Engineer [O:RV-I:BM] felt "*it's definitely more down to personalities rather than anything else*" e.g. it was accepted that junior developers could be equally dismissive to test engineers as more senior developers (in terms of both expertise and age). However, within another organisation tenure had a positive, rather than a negative impact - for example: rather than allowing any dents to their "*ego*" to fester, developers learn "*later on with their experience they understand QA are their best friend to make a successful product*" [O:CD-I:KS].

6.3.6.4 Perception of Primary V&V Interactors by Others

To some extent, the attitude adopted by developers towards test engineers is reinforced by those outside of an organisation's engineering team. Specifically, whilst some parts of an organisation might see the value in test engineers and the activities they perform, this view is not universal. As one Scrum Master and Lead QA Engineer [O:VS-I:GH] indicates: "*[t]esters to some of the business seemed to be helpful resources but to others they could not understand why we were there*". The importance of organisational support is well-known, with a lack resulting in test engineers losing both motivation and interest in their work [Perry, 2006]. V&V, both software and security-focused, is unlikely to be successful - or mature - when one of the primary interactors is considered superfluous. Unfortunately, this is not an isolated instance and reflects the view that developers are often perceived as having a higher level of social status and visibility within an organisation than test engineers [Zhang et al., 2018]. For example, one Security Test Engineer [O:RV-I:BM] (who straddles both interactor groups technically and physically) indicates, after some reflection, that developers and test engineers are not perceived as equal within their organisation, observing: "*I think there is still that kind of fundamental, developers seen as, you know, higher up than testers*". This is echoed by a Senior Test Engineer [O:DF-I:JR]: "*unfortunately developers are still perceived as being of a higher level within our organisation*". Thus it is not just developers viewing test engineers as second-class citizens [Juristo et al., 2006], but it is a view harboured throughout an organisation (we recall test engineers interact with various people within an organisation [Hass, 2008]).

Whilst test engineers appear to suffer the brunt, developers are not immune to criticism from other quarters within an organisation. For example, whilst a Head of Engineering [O:KD-I:MF] indicates that: "*[t]echnical teams have a respect for developers, as they can answer the technical questions that they might have*" (which reflects the level of knowledge held by developers and the respect that that in turn garners). It was also stated that non-technical teams, such as sales and marketing, "*have a lower level of respect*" for developers - apparently: "*[t]o them, the software has bugs and that*

is the fault of the developers". However, more generally, developers and test engineers are considered as a single group, perceived almost equal in status by those outside of engineering e.g. a Test Manager [O:RW-I:RL] indicates that whilst those outside of engineering respect the engineering team: "*I think we frustrate them sometimes*". Such frustration is primarily driven by their not: "*realis[ing] how long it can take sometimes to get even what may seem like a small feature through something*". Interestingly, the interactors external to engineering "*see the necessity for both*" developers and test engineers but "*they might lean towards developers more*". This is due to developers being able to "*do what they want and could always get it [the product] out the door without going formally through system test*" (which supports the dispensability of some test-based activities [Rodrigues et al., 2010], see Section 5.6.5). The frustration by interactors outside of engineering, regarding the time taken to progress a feature or product, is echoed by one Lead Software Test Engineer [O:RV-I:ML]. In this particular instance, marketing, where "*there would be a bit of annoyance, that stuff takes too long to happen and they always, kind of, want stuff now, now, now, now, now*" (such frustration has been witnessed within the author's experience of working for an SME). Notably, these examples highlight the competitive nature of software SMEs [Feldmann and Pizka, 2003], and how the desire in the software industry to be first to market (which shares "the spotlight with other quality attributes" [Offutt, 2002, p. 29]) can result in V&V activities being deemed unnecessary [Hutcheson, 2003].

Overall, the technical interactors, based outside of engineering, have a greater level of respect for those within engineering than the non-technical interactors. We observed examples of this above, however, we use one medium-sized organisation [O:RV] to highlight this: specifically, whilst their marketing team (a non-technical team) can become frustrated at the engineering team (both developers and test engineers) as they "*are always going to want stuff immediately*", the relationship between the organisation's engineering team and their customer support team is positive. This is primarily due to customer support being more technically-oriented. This has resulted in a relationship where they "*come over and have a chat with people around, around the engineering department*". The informal, casualness captured by this supports the positive nature of the relationship (in contrast to the energised - and emphasised by the interviewee with clicking fingers - "*now, now, now, now, now*" mindset exhibited by the marketing team above). Similarly, within another organisation, there is a close relationship between the customer support team and engineering i.e. "*[s]ervice desk worked closely with us, and appreciated our input and help*" [O:VS-I:GH]. However, and strengthening the view that non-technical interactors do not view engineers in a wholly favourable light, a Product Development Director [O:SP-I:WS] indicates whilst "*[t]he rest of the*

organisation has a high regard for the engineering team", it is felt "*some issues are over complicated by the engineering team*". Specifically, it was indicated that "*[c]ertainly some elements of the sales and marketing teams think that coding is easy and anything can be achieved that might be requested*".

Although there is a stronger appreciation for the two primary interactors by other technical interactors within an organisation, some exceptions exist. For example, one Scrum Master and Lead QA Engineer [O:VS-I:GH] indicated that the "*training team didn't like us*". Specifically: "*they would ask for new functionality and not show up to meetings to discuss, so it was developed how 'we thought best' then they would say the testers didn't test it right when they found it wasn't doing what they hadn't asked for!*" (this example also reinforces that test engineers can be seen as the easier target). Similarly, a Senior Test Engineer [O:DF-I:JR] reported that "*the support team seem to be afraid to come into the development room*". Although a small organisation, it is apparent that the "*[t]he engineering side of the company does seem to be a little isolated at times*". Both examples convey a need to improve the V&V-related communication between the different interactors throughout an organisation (which was acknowledged by the interviewee i.e. it is necessary to "*encourage more interaction with the other teams*"). The Scrum Master and Lead QA Engineer also reports that: "*[o]nce I had sat with Marketing they could see the benefits of us being there and would call to ask questions prior to visiting potential clients*". Whilst this acts as another counterexample - where we have generally seen interactors, such as marketing, viewing engineers in an unfavourable light - it also shows the value in establishing communication between the technical and non-technical interactors when it comes to V&V, which is important when considering the power held by the interactors outside of engineering.

6.3.6.5 The Influence of Power Outside of Engineering on V&V

Having established how engineers (both developers and test engineers) are viewed by those outside of engineering, it becomes necessary to understand the power relationships which exist between these two groups and the impact this has on V&V. Notably, the interviews confirmed that, generally, the interactors based outside of engineering have more power. The prevalence of this is best indicated by the Test Manager [O:RW-I:RL] within a medium-sized organisation: "*I found that not only in our organisation, but anywhere I have worked*". They vocalised that such interactors were able to exert their power by applying pressure e.g. "*if they want it they want it now*". As above, we found this sense of urgency expressed within other organisations by interactors external to engineering (e.g. "*now, now, now, now, now*" [O:RV-I:ML]). We observed similar within small organisations where, for example, the Product Development Manager

[O:SS-I:RD] confirmed that: "*the overall power would sit outside of the development team*".

Unfortunately, interactors outside of engineering - with their elevated power - can negatively impact V&V (i.e. the activities predominately performed by interactors within engineering, see Section 5.6.1). For example, as communicated by the Product Development Director [O:SP-I:WS] within a medium-sized organisation: "*[l]ast minute changes in requirements*" and "*[c]hanges in priorities from Management*" impact software and security V&V. Similar changes also impact the activities within small organisations (e.g. [O:DF]). The impact of last minute changes is well known [Cohen et al., 2004] and can "severely impact testing and product stability" [Galen, 2005, p. 87], however, we confirm this within an SME context and find that it impacts a broad range of V&V activities i.e. not just software testing.

The interactors outside of engineering are also rigorous in enforcing release deadlines, leading to some V&V activities becoming squeezed. This impacts some activities more than others e.g. as a Product Development Manager [O:SS-I:RD] vocalised: "*with testing always being at the backend of the process it's always the one that does get squeezed*" (reflecting the findings of others e.g. [Nguyen et al., 2006]). This emphasises the need for not considering testing solely towards the end of a project. Such "interference" not only impacts the activities, but also the primary interactors. For example, within a medium-sized organisation, one Scrum Master and Lead QA Engineer [O:VS-I:GH] indicated that "*releases were held if work was not complete*". Exploring who enforces this pressure, they responded it was "*[t]he business sadly*". This resulted in evident exasperation, with the interviewee conveying that they were "*[f]rustrated beyond belief*", emphasising that it was "*[l]ike I wasn't being listened to*" and that "*[t]est doesn't matter!*". This echoes the finding that "[t]he loss of testing time was interpreted to mean testing was not highly valued" [Cohen et al., 2004, p. 79]. It also supports the held view that test engineers, and test-based activities, are second-class [Bertolino, 2003, Juristo et al., 2006].

Whilst such views are typically held by developers [Juristo et al., 2006], we find they are also held by interactors with the power and influence to change such attitudes (thus also supporting the observed lack of management support, see Sections 5.6.4.2 and 5.6.5). Unfortunately, the pressure to reduce V&V often originates at an organisation's board level. This makes it difficult - if not impossible - for an individual to continue their activities when faced with a deadline. Such pressure to release evidently filters down within an organisation since, although within one organisation it is the product manager who officially determines when a product is ready to be released, the Senior Test Engineer [O:DF-I:JR] interviewed stated: "*although this is often under pressure*

from the Board". In this instance, both software and security V&V are squeezed as release deadlines loom. Within another organisation, a Software Tester [O:CT-I:DS] indicates that "*when you've got sales people selling stuff*" and when "*you've got the potential to become a multi-million pound company - well, a multi-hundred million pound company - and you've got a board that are seeing the pound signs, the pressure will be on [meeting the primary functionality]*". This results in software testing being prioritised over security testing. Interestingly, and with regards to the two most senior engineering interactors within this organisation, although "*they've got the power to try and exert their influence [in terms of the security-focused activities], but, you known, the thing is at the moment we are very sales driven. We are all about land grab if you like, get the foot in the door, get some more contracts, chances are you are going to get bigger ones afterwards. And that is really where the board and the CEO are focused*".

These examples highlight an unfortunate combination - negatively impacting security V&V - namely: not only do senior interactors view the various V&V activities as being dispensable (we recall this was the primary social factor impacting V&V, followed by a lack of management support, see Section 5.6.5), but there appears to be a preference for prioritising software testing over security testing. A reduction in testing scope - due to time pressures - has been echoed by others e.g. [Gilbert, 2011]. However, having acknowledged the complexity of security V&V (see Sections 3.2 and 5.6.5) we should not overlook this aspect, especially when the Product Development Director [O:SP-I:WS], within a medium-sized organisation, observes, in terms of security V&V: "*there is too little real understanding of what is needed at the higher levels*".

6.3.6.6 Security V&V is Complex, Compounded by a Lack of Training

Security V&V encompasses a set of complex activities [Austin and Williams, 2011]. To perform such activities effectively, knowledge of a product's internal workings is required [Thompson et al., 2002, Potter and McGraw, 2004]. Developers (as opposed to test engineers) are more likely to possess this level of understanding (e.g. as became evident: "*a developer is often able to make better arguments based on the code*" [O:RV-I:SK]). However, within one organisation, developers were more likely to want to perform software V&V than security V&V when "*[g]iven the choice [. . .] because they know it [the product] inside out and don't have to think*" [O:VS-I:GH]. Whilst this shows a utilisation of their product knowledge, it also shows an unwillingness to embrace a "notoriously difficult task" [Mahmood et al., 2012, p. 22]. A Test Manager [O:RW-I:RL], within a medium-sized organisation, also observes that developers "*probably wouldn't want to do security ones and the only reason is is that it is quite a complex area sometimes and not everyone understands it*". Similarly, this attitude

is present within small organisations, with one Product Development Manager [O:SS-I:RD] unequivocal in the view that developer performed unit testing would not encompass security e.g. "*certainly the development side of the unit tests they do would not be of that complexity*". These examples show reluctance - by the interactors best placed to address security-focused testing - to embrace a complex activity. They also highlight the impact a lack of experience and training has on security V&V.

Given the evident lack of training (see Section 5.6.7), coupled with the complexity of security V&V itself - a predominately black-box approach has been taken within many of the organisations. For example, within one small organisation [O:SS], whilst white-box testing is performed, when it comes to security, the "*specific, testing around that area is very much black-box, purely because that's the way that the QA team tend to do their testing*" [O:SS-I:RD]. However, whilst we initially perceived this as an issue of training and the skill set of those involved, it would seem, according to the Product Development Manager that, in terms of the QA team: "*they're not interested about what happens in the box, they just put their inputs in and expect the required outputs to come out*". This suggests apathy, therefore, security has almost become 'someone else's problem' within this organisation e.g. the unit testing performed by the development team avoids security because of its inherent complexity, and whilst the QA team views security-based testing "*as more niche*", their security testing is of a more functional nature e.g. "*it's more broad, more system and integration type testing that the QA team do*" (this is a rather limited approach to security testing, see Section 1.3.5). Therefore, to some extent, security appears to have fallen between the cracks within this organisation.

Similarly, a Senior Test Engineer [O:DF-I:JR], within another small organisation, considers their security testing as being more "*[f]unctional and more black-box*". Exploring why, it transpired this was due to a "*[l]ack of knowledge and expertise and often from finding items by accident*". Whilst this adds further support that security testing is complex, what it also highlights is troubling - namely, finding vulnerabilities by accident. This, and examples conveying attitudes such as: "*if I am going to run the security tests a lot of them are black-box [. . .] I will just provide the results*" [O:RW-I:RL], not only support the evident, general lack of knowledge concerning security V&V, but they also help mask the underlying problem i.e. if vulnerabilities are seen as being found within an organisation, there is unlikely to be the necessary impetus to invest further in security-focused training. Fortunately, although many organisations rely on a black-box approach, some acknowledge their lack of knowledge e.g. without any specific prompting, the Product Development Director [O:SP-I:WS], within a medium-sized organisation, indicated that the organisation's "*staff would benefit from*

further specific training in this area". Elaborating, it became apparent that "*[s]o far, the vast majority of security V&V has been put in place due to the engineering team knowing it is the right thing to do*". However, this reliance on individuals knowing what the "right thing to do" is in the first place (i.e. through having "*had to investigate how to do this themselves rather than a specific training plan being put in place*") has resulted in the interviewee fearing that "*[t]his has perhaps lead to a piecemeal approach rather than a unified strategy and standard methodology*". Ultimately, and aside from further emphasising the "piecemeal approach", we find - in all but one organisation (namely, [O:RV]) - an absence of any formally recognised security V&V authority or champion. This has probably contributed towards the observed levels of security V&V outsourcing (notably, within the organisation where there is a recognised security V&V champion, the organisation does not engage in any form of outsourcing).

6.3.6.7 Outsourcing Security V&V

Outsourcing, "a rational response to competitive pressures" [Oza et al., 2004, p. 67] (such pressures are particularly evident within software SMEs [Feldmann and Pizka, 2003]), is technologically and organisationally complex [Nicholson and Sahay, 2001]. Therefore, although outsourced engineers are involved across all V&V activities (see Section 5.6.1), with an evident desire to further increase their involvement - especially regarding security V&V (see Section 5.6.2) - we found their engagement is not without difficultly i.e. aside from resource flow concerns, there are also undesired interactions. As many organisations become frustrated when their outsourcing attempts fail [Smuts et al., 2010], it is pertinent to explore these instances in more detail.

Such frustrations were emphasised by a Senior Test Engineer [O:DF-I:JR] who, when asked whether their organisation's software V&V activities should be outsourced, responded: "*[n]o, this was done previously and left the company in a near state of ruin*". Seeking elaboration, the outsourced agency (based in India) was unreliable e.g. "*the quality of the work coming back was of a poor level and the time to get things done*" was unacceptable. Whilst it was acknowledged that "*[i]t may be different with outsource companies now*", they stated "*I don't think our company would be willing to try*" again. This supports the importance of establishing, and maintaining, trust with an outsourced software company [Oza et al., 2006] (this relationship is discussed in an Indian-based outsourcing context). However, although this outsourcing experience was negative, it would not dissuade them outsourcing their security V&V since: "*[w]e have started to look into the prospect of outsourcing to specialist companies*". Notably, and aside the usual concerns (which have become heightened based on their previous experience) of "*finding the correct company and within a reasonable budget*", it shows

an acknowledgement, within the organisation, that there is a "*[l]ack of knowledge and expertise*" with regards to security V&V, leading the interviewee to state: "*I think there are experts that can find, analyse and help resolve items much quicker*" (access to technical expertise is an obvious benefit of outsourcing [Smuts et al., 2010]).

Similarly, a Product Development Director [O:SP-I:WS] indicated their software V&V "*has been outsourced in the past with little success*". Expounding on this, it became apparent that "*the product is complex and it is necessary to understand some of the ways in which the product is intended to be used in order to design appropriate tests for it*". However, in this instance, it was not only product complexity impacting the ability to successfully outsource their software V&V, it was also the strong relationship which existed between the organisation's developers and test engineers, where: "*given that the testers and developers work so closely, outsourced testing would not work*". Arguably, whilst we would consider such a healthy, productive relationship as being an exemplar, it also shows how the closeness of the relationship potentially prevents the use of outsourcing. Notably, this organisation does not currently outsource any security V&V, however, they acknowledged that "*[a]n amount of black-box security testing could be outsourced, and perhaps should be outsourced*". The interviewee made several references to the need to improve the level of security V&V within their organisation, therefore, and as above, it is unsurprising that the organisation - although unsuccessful in the past when outsourcing their software V&V - would still consider outsourcing their security V&V. However, it was admitted, based on the strength of the working practices and relationships within the organisation, that "*the ongoing testing throughout the development process would make outsourcing of this difficult*". Further, by only outsourcing the black-box element of their security testing, they are not making the best recourse to the available, external security expertise (effective security testing is more than a black-box exercise [Thompson et al., 2002, Potter and McGraw, 2004]).

The examples above support the findings in [Oza and Hall, 2005], namely, that several difficulties and challenges exist when outsourcing. Notably, Oza and Hall discuss these difficulties specifically within an Indian software outsourcing context. We highlight that of those organisations where interviews were conducted, and where an outsourced agency was employed, it was the organisations utilising Indian-based outsourced agencies who were more often to report problems. Specifically, we did not find such negative experiences being reported when using outsourced agencies based in other countries (Russia [O:CT] and Ukraine [O:RW] for example). Whilst there might be cultural implications at play e.g. frustrations due to the deference between British managers and Indian programmers [Krishna et al., 2004] (which was supported by one Software Tester [O:CT-I:DS] who vocalised that "*deadlines don't mean anything in In-*

dia", elaborating: "*yeah it will be done by then; the deadline slips, slips, slips, slips, endlessly slips*"), there are a variety of other factors which could impact the relationship e.g. [Oza and Hall, 2005, Smuts et al., 2010] (some of which were observed by the author when managing the outsourced software V&V activities for a software SME when using an Indian-based outsourcing agency). Interestingly, Krishna et al. suggest that cultural training - for both the organisation and the outsourced agency - could improve the software outsourcing relationship, however, as found, training within software SMEs is limited (see Section 5.6.7).

Aside from the difficulties observed when outsourcing, outsourcing security V&V could itself be viewed as an "undesired interaction" - specifically, with regards to reducing the level of in-house knowledge and capability. Thus, whilst outsourcing security V&V can be considered attractive (e.g. as vocalised by one Product Development Manager [O:SS-I:RD]: "*it would be nice to have the resource, the specialist resource in-house all the time, sometimes, depending on the requirement you do have to outsource*"), when posing the question should software and security V&V activities be outsourced, one Lead Integration Engineer [O:AG-I:DS] responded: "*[i]n both cases, the ideal answer is no*". Elaborating, it became apparent that the desire to retain, and further develop, V&V knowledge in-house was a key consideration to their organisation: "*[t]esting anything in-house allows an organisation to both retain knowledge within the permanent staff and reduce the risk of inside knowledge 'getting out'*". This response also reinforces how sensitive a subject security is (security concerns, regarding the use of outsourcing, have previously been raised e.g. [Fink, 1994, Ranganathan and Balaji, 2007, Smuts et al., 2010]). In addition, and echoing the view that sometimes the application of security V&V is "*a one-off activity*" [O:SS-I:RD], or only undertaken "*[a]fter each major release*" [O:KD-I:MF], the Lead Integration Engineer [O:AG-I:DS] indicated that "*it is not always realistic to employ somebody full time in such a role*". Therefore, whilst outsourcing can help address an immediate knowledge gap, we find that not only does it hinder improving knowledge levels within an organisation (e.g. resulting in a loss of technical expertise [Smuts et al., 2010] which, in the author's own experience, also hinders the ability to appropriately guide and focus the outsourced agency), but that trust issues and resource issues can surface as a result.

6.3.6.8 Conclusions

Within the first phase we presented our findings in terms of the resourcing constraints and the undesired social factors found within software SMEs (see Section 5.6.5). The interviews afforded the opportunity to further explore these. For example, within Section 5.6.1, there is a noticeable lack of developer participation in test-based activities

(both software and security-focused). The existing literature suggests that test-based activities have been viewed in a negative light [Bertolino, 2003, Rooksby et al., 2009] and that there are strained relationships between developers and test engineers [Sawyer, 2001, Cohen et al., 2004, Juristo et al., 2006, Zhang et al., 2014], however, we did not really develop an understanding as to whether such views were actually harboured within software SMEs and whether they were applicable, by extension, to the security-focused test-based activities. The interviews helped confirm that such conflict not only exists within software SMEs, but that it has a real impact on the V&V activities being performed. As conveyed by one Senior Test Engineer [O:DF-I:JR]: "*you will always get difficult individuals from both development and test*", supporting the view that conflict between developers and test engineers is inevitable [Zhang et al., 2014, Zhang et al., 2018]. It also underlies the importance of considering the surrounding social factors (such as poor team work, poor communication and lack of motivation) and not focusing solely on the technical aspects of V&V (see Chapter 3). Additionally, we found, like [Shachaf and Rosenbaum, 2009], that it was necessary to explore both the technical and the non-technical interactors which exist within a STIN (especially since the latter, although often not directly involved in the V&V activities, have a strong influence over them and the interactors actually involved).

It is also important to acknowledge - aside from how developers perceive test-based activities and how developers and test engineers work together and perceive one another - that each type of interactor has a role to play. As a Product Development Manager [O:SS-I:RD] states, whilst "*you could argue that the development team is more influential, or has more influence [...] conversely, the QA team could stop a release being delivered*". In addition, one Lead Integration Engineer [O:AG-I:DS] indicates that "*testers' voices were considered to be as important as those of developers*". Unfortunately, this does not shield the test engineers from the majority of the levelled criticisms and pressures faced (which, as the interviews attest, originate both within, and outside, of an organisation's engineering team). This is probably in part unavoidable e.g. with activities such as system test taking place towards the end of a project, it is perhaps understandable that views such as "*[d]evelopment are still perceived as the team who make it work and test more of a team that seem to hold the releases up*" [O:DF-I:JR] are perpetuated.

Continuing with the theme of time, we recall that the primary resourcing constraint identified in the first phase was a lack of time. This is supported by other empirical studies e.g. [Park et al., 2008, Larusdottir et al., 2010, Rodrigues et al., 2010]. However, the interviews enabled us to understand the driving force behind the enforced deadlines (resulting in a lack of time) through examination of the power held by the

interactors outside of engineering and their influence on the activities (e.g. the evident lack of management support translated into the activities being viewed as dispensable). The interviews also allowed us to explore the next most reported resourcing constraint which, in terms of security V&V, was a lack of training and knowledge. As observed, security V&V is a complex topic and this has implications. Specifically, whilst there was a general preference for developers to avoid test-based activities we find, given the choice, that they would prefer to undertake software test activities rather than security-focused test-based activities. Prior to the interviews, such a finding could perhaps be surmised by extending the general avoidance and dislike of software testing by developers to encompass security-focused testing as well, however, what we predominately find is that the avoidance of security-focused test activities is based on one of expertise. This supports the distinction found in the first phase in how resourcing constraints impact the two sets of test-based activities differently i.e. security-focused activities are significantly more impacted by a lack of training and knowledge.

This lack of expertise also appears to drive the outsourcing of security V&V. As found within the first phase (see Section 5.6.2), there was an increased reliance on outsourced engineers for the security V&V activities - as well as a desire to further increase their use. In most cases, this was the result of a lack of security V&V knowledge within an organisation. This was substantiated by several organisations, who having failed in their software V&V outsourcing attempts - and who were not going to undertake such a venture again - but who were still considering outsourcing their security V&V. Notably, it was not unexpected to find some references to outsourcing failures since it has been reported that half of all offshore sourcing initiatives fail [Foote, 2004]. Foote indicates that organisations adept at people management, and handling change, tend to be more successful when outsourcing. This is concerning since, more generally, we find that the V&V activities do not receive sufficient management support (see Sections 5.6.4.2 and 5.6.5). For example, within one organisation, the Scrum Master and Lead QA Engineer reported there were "*some developers in the dark ages who still wanted to develop then put it [the product] straight to live*" [O:VS-I:GH], therefore, completely ignoring test (an example supporting the findings regarding a developer's perceptions of test-based activities and test engineers). Unfortunately, the situation was compounded by the organisation's management adopting the attitude of "*oh well its just them [the developers]*". Whilst several people attempted to change this, such attempts have yet to succeed (the interviewee acknowledged that, without training, the situation would not change). Other examples demonstrating a lack of management support include the reported attitude exhibited by the VP of Engineering within one small organisation [O:AG], which was one of disinterest towards security V&V and, in terms of security

V&V practice, we recall that there is "*too little real understanding of what is needed at the higher levels*" [O:SP-I:WS]. Thus, and coupled with the observed suboptimal communication regarding outsourcing (see Section 6.3.7.4), we feel that outsourcing security V&V should not be viewed as a "magic bullet". It also shows how many of the reported undesired interactions have become entangled with one another, for example, a lack of training, a lack of management support and poor communication - all of which not only impact the V&V activities directly, but which are also critical in determining the success of any outsourcing attempts [Foote, 2004, Krishna et al., 2004, Babar et al., 2007].

6.3.7 Existing Communication Forums

Having observed how central communication is to V&V (see Section 3.2), we further explore this by discussing several identified themes.

6.3.7.1 Co-Location of the Primary Interactors

In many cases, communication between developers and test engineers was assisted by their sitting amongst each another. As one Test Manager [O:RW-I:RL] succinctly summarises: "*[w]e sit together in the office*", therefore, "*the developers and testers communication is very good*". This is echoed by one Lead Integration Engineer [O:AG-I:DS], who observed that the communication between the primary interactors directly benefited from this physical closeness - as they: "*sat opposite one another and worked together closely to discuss how functionality might be implemented and how it could be tested, including such things as test cases required*". Aside from demonstrating a positive relationship, the topics of discussion are also notable i.e. the communication not only concerns test specific topics, but also implementation. This is suggestive, within this organisation at least, that the working relationship is not only positive, but that it is actively reinforced through interactor communication which, in turn, is directly benefited through the co-location of the primary interactors.

Unfortunately, this utopia is not universal. On discussion with an equivalently positioned Scrum Master and Lead QA Engineer [O:VS-I:GH], the communication between developers and test engineers was not very encouraging - even though they are co-located. This is especially evident when it comes to discussing new functionality: "*if new functionality was being introduced they didn't want testers involved in discussions*". Notably, within this medium-sized organisation, there existed "*a them and us*" relationship between the primary interactors. Although reinforcing the traditional divide (with developers working on new functionality and test engineers testing the im-

plemented functionality), it also negatively impacts how the test-related activities are performed within the organisation, as well as the communication between the primary interactors. For example, according to the interviewee, this attitude severely impacted the communication surrounding issues encountered by test engineers - whereupon some test engineers would no longer communicate directly with the developers. Aside from further increasing the divide between the two roles, the failure to involve test engineers in discussions concerning new functionality both delays the formulation of test cases and hinders the wider circulation of product knowledge within the organisation i.e "[t]esters need to be involved from the beginning of a project's life cycle so they can understand exactly what they are testing and can work with other stakeholders to create testable requirements" [Dustin, 2002, p. 3]. Such communication barriers not only impact the V&V activities but, by potentially impacting testability, it can also impact product quality and increase the costs incurred by an organisation [Tahir et al., 2015].

However, generally, the communication between the primary interactors was positive, with this directly impacting the V&V activities being practiced. For example, within one small organisation [O:CT], the collaboration between these interactors led to the successful pairing of a developer and a test engineer, who are now actively engaged in improving the organisation's test framework. As voiced by the Software Tester [O:CT-I:DS]: there were "*discussions in the last couple of days about ways I can improve the implementation of the framework and the tests within the framework*". Not only has such communication been regular - and recent - this has resulted in the developer becoming something of a "*mentor*". This relationship would not have been possible without their being sat near one another - with good communication existing between them - and the fact that "*we all generally get on with each another*".

6.3.7.2 The Influence of Agile on Interactor Communication

To some extent, the improved integration of developers and test engineers - and, therefore, an improvement in their communication - can be attributed to the wide adoption of Agile (see Section 5.4.1). As one Head of Engineering [O:KD-I:MF] states: "*[t]he testers and developers are on the same scrum team, so they sit next to each other and work together every day*". Therefore, the relationship between the primary interactors was considered "*[v]ery good*" within this organisation. This relationship also helped ensure good communication, whether the interactors were physically co-located (e.g. "*they sit together and talk to each other all the time*"), or remote (e.g. "*we use communication tools [e.g. Skype and Slack] for offsite people*"). This shows that the lines of communication already established between the interactors continued to persist

when working remotely from one another. Unfortunately, we are unable to conclude whether there is a direct relationship between the adoption of Agile and improved communication since, within the organisation [O:CT] where the developer and test engineer are actively working on improving the test framework together, it was observed that they have "*only being doing Agile for six months or so*", with the appended qualifier that they have been "*pretending to do it for six months or so*".

Further, within one medium-sized organisation [O:RV], the benefit of adopting Agile was seen as helping achieve more frequent product releases rather than improving communication. This was primarily due to their developers and test engineers already working within small teams prior to their adoption of Agile, therefore, causing one Lead Software Test Engineer [O:RV-I:SK] to observe that they are "*not necessarily sure that it has changed the relationship between developers and test engineers*". It was also observed that they do not have a "*gated process*", which was an allusion to the 'throw over the wall' mindset. Exploring this, the interviewee stated quite emphatically that "*we don't work like that and we have never worked like that*". Interestingly, within this organisation, some teams embrace Agile more than others e.g. "*the team I work on at the moment, we have never taken an Agile approach in the past, so we're slowly trying to introduce that [...] we are in the middle of our first sprint*"; equally, they acknowledge that "*there are other teams that embrace it much more [...] they have regular stand-ups, they have tightly coupled teams*" [O:RV-I:SK].

However, within many of the organisations the introduction of Agile has helped encourage communication through regular meetings. As one Lead Integration Engineer [O:AG-I:DS] indicates: "*[t]he development team used scrum and there was regular communication between all members of the team*", resulting in "*[e]xcellent communication between all members of the team, with enthusiasm during stand-ups, retrospective and planning meetings and good day-to-day communication*". Therefore, situating developers and test engineers, within scrum teams, has probably helped improve their communication. This is reinforced by the author when reflecting upon his own experiences of working within an SME - which did not practice Agile - where developers and test engineers, although sharing the same office floor, were grouped together by discipline and, at one point, were separated by a 4ft partition (which only helped emphasise, and give physical representation and meaning to, the evident 'throw over the wall' mentality). Whilst this was several years ago and, as noted, prior to the organisation's adoption of Agile, we find this theme continues to exist within some organisations. For example, one Scrum Master and Lead QA Engineer [O:VS-I:GH], in terms of reporting defects, indicates: "*[i]f we find anything it is reported to developers who look into it. It's then recycled back to [the] test team*". This attitude is very reminiscent of the 'throw over

the wall' mentality, which was certainly reinforced on further discussion. However, what makes this interesting, is that this organisation purports to employ Agile which, on exploring with the interviewee, resulted in the acknowledgement that they "*weren't great at Agile*".

6.3.7.3 Communication Surrounding Defects and Vulnerabilities

Overwhelmingly, we find that defects and vulnerabilities are typically logged within defect tracking systems. Across both small and medium-sized organisations, *Jira* is the most commonly referenced and discussed tool in this regard (the popularity of *Jira* is evident since, within one organisation [O:CT], they have only recently moved from *Bugzilla* to *Jira* with no real reason for the transition). However, regardless of which system is used, the communication of defects and vulnerabilities within these systems can vary. For example, one Test Manager [O:RW-I:RL] indicates vulnerabilities are not tagged differently: "*we raise them as defects*". This is adopted in order "*[t]o avoid too much complexity*". Similarly, within another organisation, vulnerabilities are raised as tasks - since "*we really don't put too much store about the type - there's something in this, somebody needs to do something with it [. . .] and it needs to be assigned to the right person to do something at the right period, we don't put any store upon the fact that it's a bug, or an improvement, or a task*" [O:RV-I:SK]. Although this organisation has regular triage meetings, vulnerabilities are unlikely to be discussed since "*it tends to be things that need a wider discussion, you know, when and if they should be fixed*". This effectively precludes vulnerabilities, since for "*security issues, it's very rare that they would make it into that, unless somebody has only just come across it 2 or 3 minutes before we've started the triage, that's about the only time you would see it in there and even then, you know, people know when things need to be investigated*". This is because issues deemed severe (i.e. vulnerabilities) will typically be discussed when found, with people not waiting for the triage meeting to occur. This demonstrates the importance this organisation places on addressing security-related issues and the difference in how vulnerabilities are communicated and discussed. However, both examples show that triage is a regular practice within the organisations studied. Typically, these meetings involve both developers and test engineers, however, in some cases, they are more inclusive, with the head of engineering (or an equivalent) also actively attending and participating. The general attitude observed, as expressed by one Lead Integration Engineer [O:AG-I:DS], was that "*[i]f there were any issues to be resolved, it was discussed openly with all members of the team who might have input*".

Whilst we have identified that defect tracking tools are generally employed across the organisations studied, there are instances, when a defect is deemed "*critical*", or

a "*blocker*", that people will sometimes forgo such tools, or electronic communication mechanisms such as email. As one QA Test Lead [O:CD-I:KS] states: "*we didn't wait to go through mails if the development team is locally seated with us*". This indicates that when faced with severe defects, such as vulnerabilities, there is a preference for face-to-face communication. However, for such communication to be successful, developers and test engineers must be physically situated with one another. Whilst such communication conveys immediacy, it does prevent a complete, historic trail forming, and further, requires a good, pre-existing working relationship between the interactors to be successful. We also found, within this organisation, that the communication surrounding defects not only takes place across a variety of mechanisms, but also with differing degrees of formality. For example, ranging from the more traditional i.e. formally capturing the defect or vulnerability within a defect tracking system, to verbally discussing them during a pre-organised, regular triage meeting (where the priority and severity of the issues are discussed); as well as more informally, for example: when "*on tea breaks*" and even "*after office hours*" where upcoming releases and "*interesting bugs*" are discussed. We also found other instances of interactors using verbal communication when situated near one another, for example, one Lead Software Test Engineer [O:RV-I:SK] indicates: "*I can ask any of the developers to clarify issues for me by wheeling myself over to their desk*" - and who adds that: "*I can drag them over either to point out is something a bug or not, could something be improved in anyway, or even I just don't understand something*".

However, generally, vulnerabilities are not distinguished from defects when it comes to their communication within a defect tracking system. Specifically, in the case of both small (e.g. [O:CT]) and medium-sized organisations (e.g. [O:RW]), vulnerabilities are only distinguished from other defects through their assigned priority and severity. In some instances, they are not even distinguished to this degree - as we found within one organisation [O:VS], vulnerabilities are only communicated via "*user story number and sprint number*". We also found that practices can vary within an organisation. For example, one interviewee [O:RV-I:SK] indicates that the importance of a vulnerability is only communicated by the assigned severity (although this in itself does not distinguish it as a vulnerability), however, when interviewing another, more security-focused engineer [O:RV-I:BM] within this organisation, it transpired that they apply an explicit label indicating that there are security implications. This inconsistent practice makes it difficult to satisfy some customer demands since, as one Head of Engineering [O:KD-I:MF] states: "*[m]any of our customers insist on a penetration test report so that they can see what vulnerabilities we have and how we are addressing them*". The inability to identify found vulnerabilities, and then successful communicate them, could

potentially impact an organisation's relationship with its customers. It also reduces the ability to predict areas of code where vulnerabilities might be more prevalent (see, for example, [Neuhaus et al., 2007]). Therefore, reducing an organisation's ability to focus their security V&V efforts. This is an important consideration given security V&V is a specialised discipline and that resources with such capabilities - and their opportunities for development - are generally limited within SMEs (see Section 5.8.1). Explicitly identifying vulnerabilities within a defect tracking system (for example, through an appropriate prefix and label), was actively practiced by one large organisation that the author has worked for. Notably, such consistent practice was not observed within any of the SMEs studied, however, it was practiced inconsistently within one organisation [O:RV] (of the three interviews conducted within this organisation, only one interviewee claimed to adopt this practice, with the others not actively distinguishing between defects and vulnerabilities - this latter approach was apparently adopted by all others within the organisation).

6.3.7.4 Communication with External Interactors

Within Section 5.6.2, we observed a significant reliance on outsourcing for security V&V. Whilst outsourcing might be considered a sound strategy for some organisations e.g. providing immediate access to a level of expertise at a point when an organisation is still maturing ("*when there is nobody with the correct skillset within the organisation to perform a task*" [O:AG-I:DS]), a lack of communication between the outsourced agency and some of the primary interators involved with security V&V is unlikely to help the situation improve internally. As stated by one Scrum Master and Lead QA Engineer [O:VS-I:GH], within a medium-sized organisation, regarding security V&V: "*[t]his is outsourced, we hear nothing about it*". Within this organisation, the Development Manager interacts with the outsourced agency. The author has observed similar when working for an SME which outsourced some of its security and penetrating testing i.e. the resulting report was not circulated to the test team. This led many of the test engineers to view security and penetration testing as 'someone else's problem'.

However, not only did we find communication limited in terms of the interactors involved when utilising outsourcing, but its frequency also drops. This is best reflected when looking at the differences in communication surrounding the software V&V activities practiced (which are performed in-house) and the security V&V activities (which are outsourced) within one medium-sized organisation. As captured by the Head of Engineering [O:KD-I:MF], software V&V "*is a sprint activity so is monitored every day*"; regarding security V&V: "*we get a detailed report from the testing company*". However, it would seem that this report is intended more for developers than test en-

gineers, since the Head of Engineering indicates: "*[t]he developers need to understand the security risks in more detail*". In general, test engineers are not involved in the communication with the outsourced engineers - even if they are physically brought into the organisation - since, as indicated by one Lead Integration Engineer [O:AG-I:DS], based within a small organisation [O:AG]: "*I believe an external consultant was brought in to perform security and penetration testing*". The lack of certainty continued with their acknowledging that this "*was not the norm*" and that as "*far as I am aware, was a one-off*". This lack of communication, and awareness, led the interviewee to see the security V&V activities in the following light, namely, that: "*[t]his is not applicable to the test activities of the core test team*". A similar response was given when pursuing several other questions with the interviewee which touched upon security V&V. Interestingly, the communication between developers and test engineers was considered good within this organisation. This shows that by not involving interactors in security-focused V&V communication apathy can develop i.e. it becomes 'someone else's problem'. Therefore, there are clearly consequences associated with moving from a daily, inclusive form of synchronous communication, to an irregular - less inclusive - asynchronous form of communication when it comes to the outsourced security V&V activities.

Although much of the communication surrounding defects and vulnerabilities remains internal to the organisations, some communication - specifically that concerning vulnerabilities - does involve external interactors. This not only includes outsourced engineers tasked with finding vulnerabilities, but also customers. As one Head of Engineering [O:KD-I:MF] indicated: "*[w]e would need to communicate that vulnerability to our customers, and then address it as quickly as possible*". To a much lesser extent, some customers actively approach organisations when concerned about reported vulnerabilities. One example concerns the Heartbleed vulnerability [Codenomicon, 2014] where, within one organisation [O:RV], it was reported that "*there was an all stop while we looked into whether we were affected by Heartbleed and customers did ask us are we affected*". An investigation was performed to determine whether their products were affected, and once complete, the results concerning this vulnerability were communicated to their customers.

6.3.7.5 Communication with Universities

As universities are centres of technical expertise, it is important to understand whether SMEs attempt to harness this knowledge. Notably, there are mixed views on the effectiveness of the relationship between universities and SMEs [Hendry et al., 2000], with some considering the communication between them as being weak and marginal

[Estimé et al., 1993, Tödtling and Kaufmann, 2001]. It is, therefore, unsurprising that only one SME was found to actively communicate with a university when it comes to V&V. In this instance, a medium-sized organisation [O:RV] working with a local university with a specialist security group. Within this organisation it was emphasised that "*[w]e do take security very seriously, we do talk to specialists [at a specific group within a university]*".

Tödtling and Kaufmann indicate that the relationships between SMEs and universities "are most effective in the case of personal collaboration which is favoured by short distances between the partners" [Tödtling and Kaufmann, 2001, p. 211]. They further observe that "the region is a very important spatial level". In this instance, the organisation is positioned approximately two miles away from the university. Notably, a Lead Software Test Engineer [O:RV-I:SK] vocalised that "*we have some very close contacts, be it from either the founders of the company, some of which came from that [group], you know, and members of the company that, you know, still have attachments to the [group]*", concluding: "*so we definitely engage with them*". This also helps support the finding that such links can help in terms of recruitment [Hendry et al., 2000]. We surmise, given there are few UK-based universities offering a complete, security-focused curriculum (for a focus on security testing, see [Kreeger, 2009]), and only a handful of universities with reputable security groups, that the opportunity for SMEs to capitalise on personal, regional collaborations with such groups, is severely restricted. Whilst this does not prevent collaboration with universities in terms of V&V more generally, the fact that no organisation attempted such communication simply supports the view that security V&V is a specialised subject, requiring specialist knowledge, and that SMEs are more comfortable with their engagement with software V&V (see Section 5.8.1).

6.3.8 Resource Flows

The interviews support, and add further explanation to, several of the observed resource flows that were identified during the first phase (see Section 5.6.7). We examine each of these, and show where they impact, and influence, the interactors, as well as the V&V activities themselves.

6.3.8.1 A Heavy Workload and a Lack of People

Whilst a lack of people impacts software and security V&V (see Section 5.6.5.1), the interviews provided some additional clarity. Specifically, there was usually more work than people available within an organisation (this has previously been reported within SMEs e.g. [Tosun et al., 2009]). For example, according to a Test Manager [O:RW-

I:RL]: "*there is always a lot on the list*". A view echoed by a Software Tester [O:CT-I:DS]: "*because we've got so much on our plate at the moment*", with a Lead Integration Engineer [O:AG-I:DS] indicating the primary impact to their V&V was "*the availability of resources to perform the necessary tasks*". Elaborating, they stated that "*the team was small but there was a huge amount of work to do*".

Although there were many such examples, within both small and medium-sized organisations, this has implications, which were not lost on the Lead Integration Engineer [O:AG-I:DS]: "*the core functionality was given the highest priority, with edge cases in functional testing being lower priority*". Thus, regarding security testing, they stated: "*the fact that no security testing was performed within the core test team gives an indication to the importance placed on it in comparison with functional testing*". Similarly, the Software Tester [O:CT-I:DS] vocalised that because of their engineering team's work load: "*security's taken a back seat*". This was also found within organisations where dedicated security V&V resource existed i.e. [O:RV], where multiple teams sought the assistance of the single Security Test Engineer (both within and outside of engineering) - as captured by a Lead Software Test Engineer: "*he's involved across the board, at almost every level*" [O:RV-I:SK].

We find that several other resource-related factors further compound the situation (see Sections 6.3.8.2 and 6.3.8.3).

6.3.8.2 Multiple Hats and Multiple Roles

Employees wearing multiple hats, within SMEs, has previously been identified [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018], although its impact on V&V has not. This impact is readily apparent e.g. within one small organisation [O:CT] security V&V is "*more the focus, one of the focuses, of the architect*", however, as the interview progressed, it transpired: "*he's doing the job of three people*" [O:CT-I:DS]. The implications of this are appropriately conveyed when referring back to a task performed by the architect which remained incomplete - with the interviewee vocalising: "*it's not his fault, he's perfectly competent and able to do it, it's just that because he was so snowed he didn't finish the job . . . [resulting in something] hacked together to get things to work and it's a mess*". We found several similar examples within this organisation e.g. they only have one C# developer (therefore, no independent person exists capable of reviewing their code) and, stated in terms of one senior engineer: "*because this guy's the one running those meetings and he's doing loads of other stuff as well*" this has impacted several engineering improvements from taking place, resulting in the view that "*somethings have improved, a lot of things haven't*". However, although they still try to retain some independence between those reviewing and testing the code, instances

of employees wearing multiple hats have clearly impacted their V&V activities. For example, it is the architect who finds most of the vulnerabilities (therefore, whilst they are performing their other roles, vulnerabilities may go unnoticed) and, in terms of reviewing their C# code, it was vocalised they "*have all got a vague idea, so at the moment, it's going to be, lets do what we can, review them to the best of our ability, because otherwise we're just not going to review it at all and that's worse*". Whilst this "make do" attitude is commendable, it unfortunately leads to the prioritisation of functional testing over any form of security testing within this organisation, since "*pressure will be on [their product's primary functionality] and anything that supports that . . . [this] is where the priority lies*".

There were also other examples, mostly concerning senior employees, where individuals were found to hold several roles. For example, the Product Development Manager [O:SS-I:RD], within a small organisation, stated that the person performing the technical lead role is also the architect: "*they perform a, sort of, dual function so to speak*" (there was indication they perform a CTO role as well). Further, in one medium-sized organisation [O:RV], whilst "*each team has a lead developer*", it transpired that "*the lead developer for the web and the platform teams are the same, it's the same person*" [O:RV-I:ML]. Additionally, within a small organisation [O:CT], the Test Manager was also the Support Manager (this was found within more than one organisation e.g. one of the participants in the first phase was both the Test and Support Manager [O:US-I:GJ] within their medium-sized organisation). Notably, both roles have their own responsibilities. However, where this differs to the previous examples is that these comprise the official roles and responsibilities of an individual i.e. it is not just someone "filling in" or "helping out" as required. Collectively, these examples show that developers and test engineers are impacted by multiple hat syndrome. It is also apparent that this can impact individuals differently. In some instances this has led to individuals no longer performing their primary role e.g. within one medium-sized organisation, a Project Lead/Developer now "*does very little development*" [O:RV-I:SK]. Or, it can prevent the potential benefits of having an engineer perform two roles e.g. a developer also performing a testing function. For example, as found within a small organisation, a Software Tester [O:CT-I:DS] indicates that whilst the developers would not be happy to perform any test-related activity, if "*[g]iven the opportunity and the resource to do it, [then] yes, but we, as I say, we are so snowed at the moment, that it's a case of we've got 10 number 1 priorities*". Therefore, in this instance, developers appear to be prioritising their development work over their test work (which is perhaps unsurprising given their perception of test-related activities, see Section 6.3.6.2).

6.3.8.3 A Developer-Test Engineer Imbalance

The ideal developer-test engineer ratio has been subject to much discussion e.g. [Burnstein, 2003, Page et al., 2008, Stober and Hansmann, 2010, Deak and Stålhane, 2013, Rice, 2016]; it varies by organisation, and even by project. However, the interviews show that an incorrect balance between the primary interactors can negatively impact V&V. For example, a Head of Engineering [O:KD-I:MF] indicates their "*available resources*" impacts their software V&V activities and attributes this to: "*[o]ur ratio of testers to developers is low at the moment*". Notably, their security V&V activities are less impacted in this way, primarily due to the outsourcing of these activities. Additionally, a Software Tester [O:CT-I:DS] considers their three test engineers to five developers as being "*seriously undermanned*". As they vocalised: "*we need a certain number of testers, to cover a certain amount of work from output from developers*". Interestingly, they indicated: "*we have struggled for a long time, just me and [the Test Manager], and it took a lot of back and forth to get them [senior management] to give the budget, to free up the budget, for a contractor tester*". However, it was emphasised this was due to budget constraints "*not, you know, the actual philosophy behind how many testers you should have versus how many developers*".

Similarly, a Senior Test Engineer [O:DF-I:JR] indicated that their current developer-test engineer ratio is "*[a]bout 3 to 1*", although they believe "*[i]t should be at least 2 to 1 and ideally I would have 1 to 1*". However, unlike the example above, there was a more pessimistic tone adopted, reflecting an SMEs limited resources: "*I don't know, the company does not seem to want to invest in the engineering side of the business at present*". The reason for this tone became evident: "*last year our test team was reduced by half, although the output from development has remained the same, we now have issues with unacceptable builds being sent out*" (we recall the importance of management supporting test-based activities e.g. a lack of management support can result in test engineers losing both motivation and interest [Perry, 2006]; as we found, this was the primary social factor impacting software testing, see Figure 5.27). This shows how the developer-test engineer imbalance is not only influenced by resource flows, but how, in turn, this impacts V&V and, ultimately, extends to product quality.

6.3.8.4 Cost and Expertise Impact Tool Use

The interviews support the observed prominence in use of open source tools and in-house tool development, as well as the general lack of awareness of security-focused tools (see Section 5.6.3.1). For example, although the Security Test Engineer [O:RV-I:BM], within one medium-sized organisation, indicated they use the commercial static

code analyser, *PVS-Studio*, this is currently under a trial license. This, by itself, would suggest a sensible "try before you buy approach", however, it was subsequently vocalised that: "*there's some other ones that are free*". This echoes the findings from the first phase (see Section 5.6.3.1), as well as other studies e.g. [Andersson and Runeson, 2002, Sitnikova et al., 2007]. Namely, there is a preference, within SMEs, to use free or in-house developed tools (e.g. the Security Test Engineer indicated that they "*rewrote some external scanning scripts*"). Thus, in this instance, we view the use of a trial license as being suggestive of an organisation showing a degree of caution regarding tool investment. However, it is not just monetary resource flow constraints impacting tool use, but expertise as well. For example, a Senior Test Engineer [O:DF-I:JR], within a small organisation, indicates: "*[w]e need to implement a monthly scan of the web based product, although due to the nature of our product and the resource we have available we have struggled to find a suitable product that can analyse the product in a reasonable amount of time*". This effectively boiled down to their "*not having the full knowledge required*" and is another example showing the impact resource flows have on the use of security V&V tools in particular. Notably, resource flows impacting tool use for the software V&V activities were not nearly as often mentioned or touched upon during the interviews - which also reflects the finding from the first phase - namely, the organisations are generally happier with their level of investment in software V&V than security V&V (see Figure 5.31).

In terms of security-related tool awareness, the Security Test Engineer [O:RV-I:BM] is the most informed - and evidently relied upon - person (both regarding their own organisation and on comparison with the other interviewees, which further emphasises the need for training). For example, having independently interviewed two Lead Software Test Engineers, within the same organisation: "*[the Security Test Engineer] has, kind of, all the training, he's got all the tools, the automated tools for checking a lot of this stuff*" [O:RV-I:ML] and "*[the Security Test Engineer] can read C++ and he's also got a number of tools that allow him to make sort of security passes on that so, you know, common security risks, like, you know, buffer overflows and things like that, he has those tools at his disposal*" [O:RV-I:SK]. This supports the identified "Do Not Know" trend when it comes to security V&V-related discussions (in particular, with regards to tools, see Section 5.6.3.1). It also highlights the danger of an acknowledged "*single point of failure*" [O:RV-I:ML] i.e. without the Security Test Engineer these tools appear to become inaccessible (in terms of where they are, through to how to run them). This became evident when posing such questions to the Security Test Engineer's line manager, whereupon, they confessed: "*would they [others within the organisation] know how to configure the tools to point to a specific test environment,*

I don't know", but they indicated "*in theory everyone would have access to them [the interviewee then laughed]*" [O:RV-I:ML]. The danger of this brings to mind the saying: "a fool with a tool is still a fool". An attitude which, perhaps unfairly, is captured by one Test Manager [O:RW-I:RL]: "*[b]ut if I am going to run security tests a lot of them are black-box. And then I will just provide the results*". Which is an example further supporting the general lack of awareness, and confidence, found within SMEs when it comes to security V&V tools, as well as the subject itself.

6.3.8.5 Resource Flows and Automation

The interviews showed that automation is generally relied upon e.g. we found use of *Jenkins*, an open source automation server, within one medium-sized organisation [O:RV]; with the Product Development Director [O:SP-I:WS], within another medium-sized organisation, indicating: "*[n]ightly build processes run further tests to ensure there are no further reaching effects of the latest checkins*" and "*[t]esters check the overnight builds every morning to ensure there are no new faults introduced, or unexplained changes in the auto-test results and any found are corrected by the developers immediately*". This emphasises the value of automation, however, in this instance, the focus is predominately upon software V&V, rather than security V&V (which is considered "*more pre-release test*" in nature). The value of automating V&V is evident in terms of issue turnaround time (i.e. from identifying to then addressing an issue), as well as improving communication e.g. "*[t]he management track the validation/autotest results as part of the regular reviews of progress within any development project. There are many metrics made available using various testing tools both in-house and bought in*" [O:SP-I:WS] (an example supporting the use of both proprietary tools and those developed in-house, see Section 5.6.3.1). However, it is apparent that this value does not typically encompass security V&V.

Whilst one Head of Engineering [O:KD-I:MF] indicated that their developers "*write automation tests*", this was somewhat grudgingly (see Section 6.3.6.2 for discussion on developer attitudes towards test-based activities). Unfortunately, coupled with this, their software V&V activities are impacted by a current lack of resources. Further, their security testing occurs outside of their release schedules and, although these occur regularly, it is clearly not as regularly as running security tests against a nightly build. This interview also identified that expertise, as a resource flow (see [Meyer, 2007, Taylor-Smith, 2016]), influences the level of automation. Specifically, whilst developers are the interactors predominately performing test automation within this organisation, the Head of Engineering stated that there is a lack of detailed knowledge concerning security vulnerabilities amongst their developers. This has implications, which are

appropriately captured by a Test Manager [O:RW-I:RL] within a different organisation: "*[w]e've got the automation, the test automation, that will only, I suppose verify and validate certain things that are already know*". If developers and test engineers do not possess an appropriate level of security knowledge, they are unlikely to be able to automate the discovery of such issues.

Further, although the ability to automate V&V activities is compounded within a security V&V context (because of the added complexity [Kreeger, 2009, Austin and Williams, 2011]), expertise, as a resource flow, is also found to impact the level of test automation more generally. For example, a Software Tester [O:CT-I:DS], within a small organisation, confesses that they are "*not capable of doing*" the necessary test framework updates. Notably, many of these examples support the divergence found within the first phase, namely, that a lack of training and knowledge is more likely to impact the security V&V activities (see Section 5.6.5.1).

6.3.8.6 Impact of a Lack of Training

The first phase showed that the training received within the SMEs was limited, especially regarding security V&V (see Section 5.6.7.3). The interviews helped convey the impact of this. For example, a Senior Test Engineer [O:DF-I:JR], within a small organisation, indicated their approach to security testing is "*[f]unctional and more black-box*". Enquiring why, this was due to a "*[l]ack of knowledge and expertise and often from finding items by accident*". Similarly, a Software Tester [O:CT-I:DS], within another small organisation, indicated that whilst "*security is the job of everybody, not necessarily everybody knows enough about it to contribute*". As observed, a black-box approach, and a focus on a product's security mechanisms (i.e. a purely functional perspective) is generally considered insufficient [Thompson et al., 2002, Potter and McGraw, 2004].

The inherent complexity of security V&V was acknowledged on discussion with the Test Manager [O:RW-I:RL] within one medium-sized organisation; and, although it was vocalised: "*I think we just need resources really, you know, more human resources*", it was emphasised that they also needed to be adequately trained (a lack of training and knowledge is known to impact security V&V [Larusdottir et al., 2010, Austin and Williams, 2011]). However, although this organisation is somewhat resource constrained at present, on attempting to understand whether support was available within the organisation to improve, a pragmatic view was taken e.g. "*[t]hey [management] do listen and they are aware that, you know, we need a few more resources [...]. The way I kind of look at myself, we are an expensive resource, we might add value, but we also cost them a lot*". Although this interviewee has worked for both SMEs and

large organisations, it shows how V&V is adversely affected by constrained resource flows i.e. not only by a lack of engineers in general, but also by a lack of training and knowledge. Further, and similar to developers lacking detailed knowledge of vulnerabilities (as found within one medium-sized organisation [O:KD]), we find, within another medium-sized organisation [O:RV], that this is exhibited by test engineers as well e.g. "*if someone asked me to list vulnerabilities I would just be parroting back, you know, buzz words that I'd heard of*" [O:RV-I:SK].

Whilst the examples above have predominately focused on a specific lack of security knowledge, we should not overlook the knowledge required to perform effective security V&V - namely, coding and testing expertise as well [Kreeger, 2009] - as a lack of knowledge and training in these areas also impacts the security V&V activities. For example, one Lead Software Test Engineer [O:RV-I:SK], within a medium-sized organisation, indicated that their organisation's test engineers are not involved in code reviews (which is generally the case, see Figure 5.12; however, we recall their value as a means of communicating key information [Coram and Bohner, 2005], enabling people to build up the knowledge necessary to avoid treating software as a black-box). Although their product is predominately developed in C++ (which the interviewee considers "*quite a heavyweight programming language*"), the interviewee is more comfortable with Python and Perl. However, the absence of test engineers from code reviews, within their organisation, is attributed to the developers and test engineers often possessing "*a different skill set*" (which has been recognised by others e.g. [Dhaliwal et al., 2011]). According to one Test Manager [O:RW-I:RL]: "*I'm not a psychologist or anything, but it almost appears that brain functions are different in those two types of people*", leading them to acknowledge that "*it is very difficult to find someone who can do both*" (the author's experience hiring software engineers supports this view). This is further compounded when factoring in security knowledge as well. Notably, the Test Manager attributes this to a lack of training, stating: "*I think it is just a knowledge gap. That, if you look at development and test, most of them would understand what each other does. I am not saying coding and things like that, or understanding testing practices, but they could probably cross over to a point, and when you sit in a meeting everyone understands it. But when I am talking about security, I'm saying if you are going to a meeting and ask, or talk about fault injection, or trying to run Java scripts on pages, not everyone gets it, yeah. And I think that's probably why. I think if you look at an average tester, and I am talking about the test perspective coming in now, they all, the experienced ones will know how to test, but they won't all know how to do security testing*".

This lack of in-house knowledge can result in outsourcing.

6.3.8.7 Expertise and Outsourcing

Whilst outsourcing brings several advantages, addressing an organisation's knowledge gaps has been emphasised [Ikerionwu et al., 2017]. Therefore, that we found significant reliance upon outsourced engineers - in particular for the security V&V activities (see Figure 5.12), as well as a desire to further increase this reliance (see Figure 5.14) - it becomes essential to understand why these levels of utilisation and reliance exist. The interviews helped provide some indication. For example, the Lead Integration Engineer [O:AG-I:DS], within a small organisation, indicated: "*[o]f course, in an ideal world, a software tester with knowledge and experience of security testing would have been involved, but no such person existed within the organisation*". Such an attitude explains why some organisations turn to the use of outsourcing (especially in areas of complexity, when it becomes necessary to "take advantage of vendors' technical expertise" [Dey et al., 2010, p. 93]). As an example, a Scrum Master and Lead QA Engineer [O:VS-I:GH] indicates that their medium-sized organisation, in terms of bringing their security V&V activities in-house i.e. to remove their current reliance on outsourcing, "*is not ready to do that*". This is due to resourcing constraints e.g. "*it would involve training the dev and test people*" (training opportunities within small organisations are limited [Basri and O'Connor, 2010], see Section 5.6.7.3). We also find, on discussion with a Head of Engineering [O:KD-I:MF], based within another medium-sized organisation which employs an outsourced agency for their security V&V activities, that "*[t]he developers need to understand the security risks in more detail*". Specifically, "*they get a report from the testing company, and have to research the vulnerabilities to understand their meaning and the solution*". Although it was indicated that "*[s]upport is available to do this*", it is an interesting example further highlighting the security-related knowledge gap which exists within software SMEs i.e. whilst the security V&V activities themselves are outsourced to address a knowledge gap, it is apparent, at least in this instance, that understanding the resulting communication is also problematic and not without issue.

Aside from expertise, it is important to not overlook the impact an SMEs resource flows have on the use of outsourcing more generally. For example, within some of the organisations, resourcing constraints impact the use of outsourcing even when there is an acknowledged lack of security V&V expertise within an organisation, as well as an acknowledged need to perform the activities e.g. the Lead Integration Engineer [O:AG-I:DS], within a small organisation, indicates that: "*partly because of a lack of resources available within the test team and the cost of performing security testing outside the main organisation*" is the reason that no security testing was performed, although a "*[h]igh importance was placed upon it but the functionality of the system was the pri-*

mary concern". Similarly, within another small organisation, a Senior Test Engineer [O:DF-I:JR] indicates that although they have started to investigate the outsourcing of their security V&V, the difficultly resides in finding a company "*within a reasonable budget*". This example shows how several identified resource flows are entangled e.g. the interviewee acknowledges that outsourcing provides "*experts that can find, analyse and help resolve items much quicker*", thereby addressing the expertise resource flow, however, this is, in turn, impacted by a monetary-based resource flow.

6.3.8.8 Resource Flows and Release Deadlines

We find that the resource flows surrounding the V&V activities can change when a vulnerability is found. For example, a Senior Test Engineer [O:DF-I:JR], based within a small organisation, indicates that the level of support given to addressing a vulnerability increases when such an issue is discovered (this is in comparison to when non-security-related defects are found). However, this support is not guaranteed, since both software and security V&V activities become squeezed within this organisation as release deadlines loom (there are various factors which can cause this e.g. being first to market [Nguyen et al., 2006], or to reduce the impact of delays encountered elsewhere on a project [Bashir and Goel, 1999]). Further compounding the situation, from a security V&V perspective, we find, when given the choice either to perform software testing or security testing, the general preference is to prioritise software testing in order to meet release deadlines. According to one Software Tester [O:CT-I:DS]: "*you probably find that answer quite a lot*". It is evident this decision is influenced by resource flows, for example, the complexity of security V&V, coupled with a lack of training, results in the primary interactors generally prioritising software testing, see Section 6.3.6.6.

In a medium-sized organisation, the Scrum Master and Lead QA Engineer [O:VS-I:GH] indicated that finding a vulnerability would not hold up a release (thereby not incurring any additional resourcing cost), as they adopt an "*it'll be fixed and patched*" attitude e.g. they make a release, develop a fix, test the fix and then release the patch (this "Sell First, Fix Later" approach is relatively common [Arora et al., 2006, p. 465]). However, whilst this does not impact the current release, it does impact subsequent ones e.g. "*[w]e would lose a developer and tester to get the fix out*", notably: "*[r]egardless of where the sprint was*". Such issues can distract developers (and, as we have found, test engineers as well) from making the next release on time [Hartman, 2002]. In contrast, a Lead Software Test Engineer [O:RV-I:ML], also within a medium-sized organisation, discussed delaying a release: "*if something hasn't been fully tested by the end of a sprint it'll fall over to the next one*". Whilst this will delay a release, it is supported by the organisation: "*the company doesn't really have much of a problem with that.*

It may annoy some people, but there's kind of an acceptance that it's better to release something late than broken". This reinforces the need for the primary interactors, and the V&V activities, to have the support of their wider organisation (see Section 6.3.6.5 for the impact interactors outside of engineering can have on the activities when release deadines loom). However, if something is considered "*trivial*": "*we'll launch with that bug there, we know it's there, but it's not going to have a big enough impact, you know, the risks are very low, so we'll launch with it there and fix it at a later date*". Similarly, the Product Development Director [O:SP-I:WS], within a medium-sized organisation, indicates "*[i]t would depend who the release was mainly aimed at*", elaborating: "*[c]ustomers hosting the product fully inhouse, behind their own firewalls, would be less affected than those using the product in the cloud. But certainly, if there was a risk of a security breach the release would be postponed until the vulnerability had been plugged*".

Generally, when it comes to vulnerabilities, we find a preference to delay a release so that either the issue can be addressed (e.g. "*if there is an issue we will delay the release and release at the appropriate time*" [O:SS-I:RD]), or so that it can be sufficiently evaluated to gauge its impact. Notably, the investigation, and the resulting categorisation of a vulnerability (which is compounded by both subject complexity and a lack of knowledge and training, see Sections 6.3.6.6 and 6.3.8.6), is also going to delay a release. This is especially the case when an organisation needs to: "*get the necessary people involved and make the right decision*" and achieve "*a final group go or no go*" [O:RW-I:RL], since conflict between engineers, and those outside of engineering (i.e. "*the guys at the top hold the purse strings*" [O:CT-I:DS]), is inevitable (conflict is a natural consequence of interaction [Singleton et al., 2011]). This reflects an acknowledgement, within SMEs, of the costs which can potentially result when releasing a product found to contain a vulnerability (see Section 1.3.4). However, as we have seen, there are also examples, albeit fewer in number, of the "Sell First, Fix Later" approach (e.g. [O:VS] and [O:KD], both of which are medium-sized organisations), which is probably a reflection of the competitive nature of SMEs and the software market [Fayad et al., 2000, Feldmann and Pizka, 2003].

6.3.8.9 Conclusions

Whilst the first phase allowed us to observe the effect of resource flows, the second provided us with some deeper insights e.g. although we found several V&V activities were never performed within some of the organisations - with this being attributed to resource flow constraints - the second phase showed that this was due to a variety of factors, not just an overarching "lack of resources". For example, many interactors

(typically senior employees) frequently perform multiple roles, often resulting in V&V tasks being impacted - in particular, security V&V activities. We also found that a developer-test engineer imbalance leaves some interactors "swamped" and thus focusing on a product's primary functionality, in turn, leading to security V&V becoming deprioritised (as expressed by one interviewee: "*security's taken a back seat*" [O:CT-I:DS]). A lack of human resources also influences when the V&V activities occur within an organisation's software development lifecycle. We recall the increasing costs to address defects and vulnerabilities as a project nears completion (see Section 1.3.4).

Further, the impact of interactors wearing multiple hats, or performing multiple roles, has greater impact on an organisation's ability to find vulnerabilities than defects in general. This is the case within small and medium-sized organisations e.g. within one small organisation [O:CT] it is the architect who finds most of their vulnerabilities, but this individual performs three roles; similarly, within a medium-sized organisation [O:RV], the Security Test Engineer finds most of the vulnerabilities within their products (this was independently affirmed by their line manager), however, they are "*involved across the board, at almost every level*" [O:RV-I:SK]. Notably, these examples concern individuals, affected by resource flow constraints, ultimately impacting their organisation's ability to find vulnerabilities. However, more generally, it is a lack of training and knowledge which impacts the ability to find vulnerabilities, leading other interactors either "*finding items by accident*" [O:DF-I:JR], or a suboptimal state of: "*we may find, kind of, stuff [vulnerabilities] occasionally, but we don't have a particular mantle of security training*" [O:RV-I:ML]. It thus becomes evident that security V&V activities, such as security testing, are perceived as being "*more of a specialisation*" [O:RW-I:RL]. A view supported by the reliance one organisation [O:RV] places upon the sole Security Test Engineer. For example, one Lead Software Test Engineer [O:RV-I:ML] states: "*I would do a little [security testing], but it's kind of very rudimentary stuff, it's checking everything is sent over SSL, checking certificates are valid. Very basic XSS testing. But most of it would, the vast majority would be left for [the Security Test Engineer]*". With another Lead Software Test Engineer [O:RV-I:SK], within the same organisation, indicating: "*anything security-related just goes to [the Security Test Engineer]*". Because of this it was observed that "*it's important for us to, sort of, identify those test tasks, early enough for him*" [O:RV-I:SK] (which reflects both an acknowledgment of the increasing costs to address security-related issues as a project progresses, as well as the need to ensure that this lone "specialist resource" is available to assist).

Unsurprisingly, the Security Test Engineer, according to their line manager, "*is kind of a single point of failure, he knows an awful lot, he's very experienced, but he*

is a single person, and there are limits to what he can get done" [O:RV-I:ML]. This was also borne out when interviewing the Security Test Engineer. Notably, aside from this Security Test Engineer, there was a complete absence of interactors dedicated to security V&V within any of the organisations where the interviews were conducted (thus confirming the finding from the first phase, see Section 5.8.1). Coupled with the lack of in-house specialists, we found, in general, a view of security V&V echoing: "*I certainly haven't got the [security] knowledge, [the Test Manager] sure hasn't got the knowledge*" [O:CT-I:DS]. This was readily apparent, and continually reinforced, throughout the majority of interviews - thereby emphasising the inherent complexity of security V&V [Kreeger, 2009, Austin and Williams, 2011]. This complexity, and the associated general lack of familiarity within the organisations as a whole - compounded by a lack of training (see Sections 5.6.7.3 and 6.3.8.6) - has led many to outsource these activities (see Section 6.3.8.7). Significantly, the Security Test Engineer's organisation adopts a "*not for anything*" [O:RV-I:SK] approach to outsourcing, appearing to want to retain, and develop, expertise within the organisation itself. Whilst several undesired interactions were found to surround the outsourcing of security V&V (see Section 6.3.6.7), some organisations have successfully embraced the practice. For example, one small organisation [O:CT] outsources some of their integration work, employing four Python developers (however, review and test-based activities are performed by employees within the organisation, thus retaining some level of in-house knowledge and expertise).

The interviews also echoed another finding from the first phase, namely, the impact of time (which was the most frequently reported resourcing constraint across all of the activities, see Section 5.6.5.1). Notably, the Product Development Manager [O:SS-I:RD], within a small organisation, indicated that "*time's the key one*" in terms of what impacts both their software and security-focused V&V activities. As they observed: "*with testing always being at the backend of the process it's always the one that does get squeezed*". Whilst this attitude is supported by the existing literature e.g. [Bashir and Goel, 1999, Nguyen et al., 2006], it was conveyed by the interviewee with a fair degree of acceptance. They also vocalised that: "*as with all projects, if a project has a budget that you have to, kind of, operate within, that's, kind of, the main driver*" (which also reflects the finding from the first phase concerning a lack of budget, see Figure 5.26). The organisation attempts to mitigate the impact of these resourcing constraints (both time and budget) by adopting a risk-based approach e.g. they examine "*the risk around say a particular area, in terms of the level of assurance that needs to be done, in relation to what the function, or that component would do, or what relies on that component*". However, inevitably, any reduction in time will correspond with a reduction in test

scope [Gilbert, 2011] and, invariably, we find this is likely to lead to the prioritisation of software test-based activities over security-focused ones (see Sections 6.3.6.5 and 6.3.8.8).

Significantly, the impact of time constraints, and release deadlines, also impacted software improvement activities. This echoes the findings of others e.g. [Rautiainen et al., 2002], however, we find this encompasses security V&V as well. For example, although the Security Test Engineer [O:RV-I:BM] indicates "*there is definitely time that I have to try and, you know, try different things*", they also stated: "*like I said before, my todo list has, try out this tool and obviously when there's a release on, I'll be, you know, those will be bumped off*". This supports the view that, within small organisations, people "are always busy with their daily routines" [Wong and Aspinwall, 2004, p. 54] and, notably, that none of the primary interactors are immune to the impact of release deadlines e.g. the Security Test Engineer vocalised that "*if marketing came along and said there's a deadline for next week and this has to be going out right now, then I would have to drop everything and, you know, help with that*" (an example also conveying the power held by interactors outside of engineering, see Section 6.3.6.5 for relevant discussion). Notably, such time constraints can also impact an individual's ability to develop their own security V&V skills e.g. on discussing a hypothetical example, concerning the development of a fuzzing tool, the Security Test Engineer indicated that whilst "*I'd like to think that anyone would be given the chance to do it, especially if it's going to help them learn more*", they acknowledged: "*realistically*" it would go to the person capable of completing the task in the shortest time.

In summary, many of the resource flows encountered are inherently entangled, with a Lead Software Test Engineer [O:RV-I:ML] concluding that it is a "*combination of everything really*" which impacts their security V&V activities. Notably, security V&V activities are more susceptible to resourcing constraints - as summarised by one Software Tester [O:CT-I:DS]: if a choice has to be made between applying software V&V or security V&V on a project, software V&V will be prioritised (with the interviewee stating: "*you probably find that answer quite a lot*"). This attitude is also reflected, more generally, when observing the average levels of expenditure, allocated per activity, on a typical project (see Figure 5.30) - thereby demonstrating a prioritisation of resources, within SMEs, in order to perform the software V&V activities.

6.4 Conclusions and Reflections

Having completed the second phase of data collection and analysis, we now show how this phase supports, and extends, the findings from the first phase (Chapter 5), thus addressing the research objectives.

6.4.1 Software and Security V&V Practice

Whilst the first research objective involved examining how security V&V is practiced, supported, and perceived, within UK-based software SMEs, the third research objective encompassed establishing whether such organisations can possess a mature security V&V practice without a correspondingly mature software V&V practice. As before, we structure our conclusions firstly by interactor analysis and then by network relationship analysis.

6.4.1.1 Interactor Analysis Summary

The interviews confirmed the diversity of the human interactors, directly and indirectly, involved across the V&V activities (thereby showing many of these activities are, inherently, organisationally cross-cutting [Mäntylä et al., 2012]). These interactors range from dedicated, technical resources (e.g. developers and test engineers), to non-technical resources with a vested interest in the activities (this can have both a positive and negative effect on V&V, see Section 6.4.1.2). Thus, aside from the previously identified primary and secondary human interactor groups (see Sections 5.6.1 and 5.6.2), the interviews also identified an additional group - namely, the tertiary human interactors (see Section 6.3.3). This latter group encompasses interactors not directly involved with the V&V activities, but who hold influence over them and the interactors performing them (e.g. controlling resource flows, see Section 6.3.8). It was important to include these interactors within the V&V socio-technical network, and to further explore them, since these, primarily non-technical stakeholders, can impact the motivation levels of those performing V&V (for example, see Sections 6.3.5.3 and 6.3.6.5) and, ultimately, control the resource flows surrounding the activities (see Section 6.3.8).

Further, the interviews also provided some structural context to the primary interactors and confirmed their traditional role divide (i.e. the activities they were more likely to embrace, see Section 6.3.1). Although the interviews helped corroborate the findings from the first phase, regarding the human interactors involved with the V&V activities, there was one notable exception: we found, within one medium-sized organisation [O:RV], the presence of a dedicated security V&V resource (i.e. the Security Test Engineer, see Section 6.3.1). However, notwithstanding this exception, the inter-

views confirmed the overwhelming absence of such roles, as well as the limited recourse being made to internal security expertise more generally within an organisation. This also supports the previous finding regarding the reliance SMEs place on outsourcing their security V&V activities (see Figures 5.12 and 5.14). Whilst we found a degree of acknowledgment that bringing some of these outsourced activities in-house would be preferred, an organisation's resources would often prevent this (see Section 6.3.8.7). Generally, when discussing outsourcing, this usually concerned security V&V, which supports the view that the activities are complex and require specialised resource.

We also found, throughout the interviews conducted, that the interactions and lines of communication between the various types of interactor were richer than first expected (thus confirming that V&V, and software development, are not typically performed by individuals in isolation [Aurum et al., 2002, Mosley and Posey, 2002, Tervonen and Harjumaa, 2004, Yilmaz and Phillips, 2006, Takanen et al., 2008]; which, in turn, emphasises the importance of adopting a socio-technical perspective when studying V&V, see Chapter 3). Notably, this level of interaction is not restricted to the confines of either a specific interactor group or type (e.g. test engineers only interacting with other test engineers and only occasionally with developers), but that interactor interaction is more pervasive and encompassing. This supports the view that V&V involves interactors who communicate with many different people, both within their organisations and externally [Hass, 2008]. Typically, such interaction and communication concerns the software V&V activities (which is to be expected, given the evident dearth of knowledge surrounding security V&V, see Section 6.3.6.6).

The interviews also helped highlight the diverse set of non-human interactors that exist within the V&V socio-technical network (i.e. the tools and processes used by the interactors; however, unlike with the human interactors, the interviews did not identify any additional categories of non-human interactors, see Section 6.3.4). Whilst several specific tools were referenced by the interviewees, we typically found a repeat of the observed reliance on open source tools, or tools developed in-house (see Section 5.6.3.1). The interviews also supported the identified trend of interactors appearing less confident when discussing security V&V tools (see Section 6.3.8.4). Similarly, the interviewees were less certain about whether their organisation followed any form of secure development process e.g. with interviewees indicating: "*I don't believe so*" [O:AG-I:DS] and "*not that I am aware of*" [O:RW-I:RL]. When a secure development process was found to exist, in many instances, this was attributable to an initial external influence e.g. through meeting a particular certification or through fulfilling a customer request (see Sections 5.6.4.5 and 6.3.5.4). Notably, the organisation with the greatest level of awareness, and maturity, regarding the tools and processes supporting

the security V&V activities, is the organisation [O:RV] with the Security Test Engineer (who helped introduce and formalise many of their adopted tools and processes). However, although this individual has been critical in maturing their organisation's level of security V&V practice, we find, even within this organisation, that their adoption of a recognised secure development process was initially driven by a customer request, supporting the view that smaller software organisations do not have an ingrained process culture [Valtanen and Ahonen, 2008].

Having considered the interactors, we now examine their incentives and motivations for performing V&V. We recall, from the first phase, that improving software quality was the primary motivator for performing any V&V activity. Whilst this was the case for software and security V&V, it was notably more prevalent in terms of software V&V. We ascribed this to the interactors not affording importance to security as a quality attribute. However, pursuing this view with several interviewees, we found, throughout their organisations, an acknowledgment of the importance of releasing a secure product. The desire to improve software quality, in terms of security, was driven by both intrinsic and extrinsic factors. For example, interactors were often motivated on an intrinsic level e.g. having "*personal pride in their work*" [O:VS-I:GH] and acknowledging that "*whatever I do in my job has a direct impact on someone else*" [O:RW-I:RL]. However, extrinsic factors also existed e.g. to avoid the costs associated with reputation damage (with some interviewees echoing several of the reasons captured within Section 1.3.4).

Therefore, whilst the interviews helped convey that security is perceived as a quality attribute by many of the interactors (e.g. a Product Development Director [O:SP-I:WS] stressed that security "*is seen as an attribute with equal importance with others*" and one Head of Engineering [O:KD-I:MF] viewed it as sitting "*alongside performance and accessibility*"), we find security has often "*taken a back seat*" [O:CT-I:DS]. Notably, extrinsic factors of motivation were more likely to galvanise interactors into performing software V&V than security V&V. This is unsurprising given the general lack of processes covering these activities (see Sections 6.3.4.2 and 6.3.5.2), as well as the limited management support afforded in general (see Section 6.3.5.3). The situation is further compounded by the undesired interactions and resource flows surrounding the security-focused activities in particular (see Sections 6.3.6 and 6.3.8). However, the interviews suggest that customers are becoming more influential e.g. within one medium-sized organisation "*customers insist on a penetration test report*" [O:KD-I:MF], and were responsible for instigating the adoption of a secure development process within another medium-sized organisation [O:RV]. This "awakening" of customers was observed by some of the interviewees e.g. "*externally, the whole technology sector is becoming more, you know, aware of security and focusing on security, so, hence why the sales*

guys are getting those questions now and all the support tickets coming in and asking about our security procedures and how we work, so there is also an external push as well" [O:RV-I:BM], as well as others e.g. [McGraw, 2006, Asghari et al., 2016, Owsley, 2017, Zabicki and Ellis, 2017]. This demand is evident in the number of organisations either meeting, or striving to meet, particular regulations and certifications, which is often driven by the need to sell a product to a specific market e.g. "*you can't make sales to certain parts, to certain industries, if you aren't certified*" [O:RV-I:SK]. This supports a finding from the first phase, namely, interactors are generally more motivated to perform security V&V by external influences (see Section 5.8.1).

In summary, the interviews confirmed that security V&V is performed less frequently than software V&V. For example, we observed a focus on a product's primary functionality and, whilst software V&V occurred regularly throughout the software development lifecycle, security V&V, when performed, would often occur outside of release schedules. We also found, when given a choice, interactors (including the primary interactors) would typically opt to perform software V&V activities. This supports the findings from the first phase (see Section 5.8.1). In addition, interviewees exhibited a lack of confidence when discussing the security V&V being performed and the subject matter itself. For example, we often found a high-level understanding being demonstrated by most of those interviewed when discussing security V&V and vulnerabilities - in the words of one Lead Software Test Engineer, they would just be "*parroting back*" [O:RV-I:SK]. Therefore, as with the first phase, we conclude that SMEs are much more active, engaged, and confident, with their software V&V activities.

6.4.1.2 Network Relationship Analysis Summary

The second phase also permitted deeper exploration of the relationships within the V&V socio-technical network, providing greater insight into how the primary interactors interact with one another and how they participate in the various activities. For example, in terms of communication, developers and test engineers were often co-located with one another (see Section 6.3.7.1). This was found to assist the communication between the two primary interactors, with one Lead Software Test Engineer [O:RV-I:SK] vocalising that communication with other engineers was often initiated by "*wheeling myself over to their desk*", as well as being able to "*drag them over*" to discuss issues encountered. Whilst it would be tempting to attribute this, and other such examples, solely to the adoption of Agile (which actively promotes communication [Hazzan and Dubinsky, 2008, Overhage et al., 2011]), within many of these organisations the primary interactors already sat closely together prior to their adoption of Agile. It also became apparent that several of the organisations, both small and medium-sized,

who have adopted the Agile mantle, are either "*pretending to do it*" [O:CT-I:DS] or working in an "*Agileish*" [O:RV-I:ML] way. However, we did find that the adoption of Scrum - which necessitates regular meetings e.g. daily stand-ups - has actively helped promote lines of communication between the primary interactors, thereby supporting the view that they facilitate communication amongst the entire team [Chau and Maurer, 2004]. To a lesser extent, we found that such meetings can help address cultural barriers and disparities between interactors [Sutherland et al., 2007] e.g. within one small organisation, where Scrum was used, the Lead Integration Engineer [O:AG-I:DS] indicated: "*there was regular communication between all members of the team*", whereupon "*[d]evelopers and testers sat opposite one another and worked together closely to discuss how functionality might be implemented and how it could be tested, including such things as test cases required*".

However, the co-location of the primary interactors is not a panacea. Notably, the communication between the interactors, in one medium-sized organisation, was considered as being "*[o]kay at best*", with the Scrum Master and Lead QA Engineer [O:VS-I:GH] evidently frustrated in terms of the broken lines of communication: "*[i]t always bothers me when those with a vested interest don't fully participate in the process, even if it's just attending meetings. You wouldn't build a new home without seeing plans and speaking to people*". Similarly, a Senior Test Engineer [O:DF-I:JR], within a small organisation, indicated: "*[c]ommunication between developers and testers can always be difficult*". This was especially apparent when discussing defects and vulnerabilities (echoing the conflict observed by others e.g. [Cohen et al., 2004, Vogel, 2011]). For example, the Senior Test Engineer elaborates that "*developers often put an issue back to test just stating fixed, which isn't of help, also testers can be guilty of not supplying enough information*". They also indicated that the situation is further compounded by relying on defect tracking systems as a means of communication (however, they note: "*[a]lthough we are encouraged to speak to each other using email/phone and instant messenger, to try and ensure that an issue doesn't spiral for no reason*").

As found within the first phase (see Section 5.6.6), the interviews show that the interactors utilise a variety of synchronous and asynchronous forms of communication. This is particularly apparent when interactors discuss defects and vulnerabilities since, although they are typically logged and communicated within defect tracking systems, they are also discussed in various face-to-face contexts (e.g. triage meetings and daily stand-ups). However, within the employed defect tracking systems, we find that vulnerabilities are not actively distinguished from defects i.e. they are all considered defects. Whilst vulnerabilities come from defects [McGraw, 2006], we acknowledge that some are more important than others [Schechter, 2002] (vulnerabilities, in particular, for the

reasons stated in Section 1.3.4). The failure to represent, and communicate, a vulnerability as being more than a defect can lead to vulnerabilities becoming 'lost in the noise' e.g. finding unlabelled vulnerabilities amongst defects can be challenging [Peters et al., 2017], thus organisations should label them appropriately [Russell and Van Duren, 2016]. Ultimately, the discussion and communication of vulnerabilities - regardless of mechanism - is compounded by their complexity e.g. when is a defect a vulnerability and do several defects equate to a single vulnerability or multiple vulnerabilities [Ozment, 2006]. In addition, both developers (e.g. "*developers need to understand the security risks in more detail. Currently, they get a report from the testing company, and have to research the vulnerabilities to understand their meaning and the solution*" [O:KD-I:MF]) and test engineers (e.g. "*I would just be parroting back, you know, buzz words that I'd heard of*" [O:RV-I:SK]) struggle in terms of the inherent complexity of identifying and communicating vulnerabilities.

The importance of ensuring good, clear lines of communication between the primary interactors (especially when discussing defects and vulnerabilities) becomes apparent when examining the relationship between the two interactors more generally. For example, whilst the interviews helped confirm that test-based activities are general perceived as being second-class activities [Bertolino, 2003] by developers - with one Product Development Manager [O:SS-I:RD] equating the activity to that of "*washing the dishes*" - we find the situation less black and white. Certainly, within some of the organisations, this varies by developer e.g. "*some are really interested and want to [be] involved in test-related activities*" [O:CD-I:KS]. However, even with the wide adoption of Agile (see Figure 5.4) - developers continue to view test-based activities negatively. Further compounding the situation, we find that whilst developers are best placed to view software as being more than a black-box (which we recall effective security testing requires [Thompson et al., 2002, Potter and McGraw, 2004]), they are more likely to perform software testing than security-focused testing. Once again, this varies by individual developer, however, generally, there is a preference to not perform any test-based activity and then, only if required (e.g. because it is a part of their official responsibilities or is the result of a management dictate or process) preferring to perform software testing.

Whilst the prevalent adoption of Agile should have helped increase the level of cohesiveness between team members (encompassing both developers and test engineers), this was not evident within many of the organisations. For example, we find that "*a them and us*" [O:VS-I:GH] attitude still exists (which is supported, even when organisations adopt Agile practices, by many of the case studies discussed in [Watkins, 2009]). We also find that the adoption of Agile has not helped with the distribution of V&V activities between the primary interactors (i.e. developers predominately continue to

focus on design and code-based activities and test engineers on test-based activities, see Sections 5.6.1 and 6.3.1). This effectively counters the aspired for generalist [Meszaros and Aston, 2007] and the blurring of roles [Crispin and Gregory, 2009]. Once again, we attribute this to the evident "*pretending to do it*" [O:CT-I:DS] and lackadaisical approach to Agile (team agility can only be achieved when a team buys into Agile principles [Watkins, 2009]). Ultimately, this has helped perpetuate the attitude of developers viewing test-based activities, and test engineers, as being second-class (see Sections 6.3.6.2 and 6.3.6.3, as well as [Bertolino, 2003, Juristo et al., 2006]).

Notably, the interviews not only helped identify a third set of interactors (see Section 6.3.3), but showed that these interactors also perceive test engineers as being somewhat "lower down" than developers (this was apparent in both small and medium-sized organisations, see Section 6.3.6.4). This is a significant finding as much of the reported diatribe surrounding test engineers has been attributed to developers (e.g. [Bertolino, 2003, Cohen et al., 2004, Juristo et al., 2006, Watkins, 2009, Zhang et al., 2014]). Whilst we are not suggesting that a similar level of conflict towards test engineers originates from the other interactors within an organisation, the interviews suggest that test engineers are less likely to be treated as equal to developers e.g. "*[t]esters to some of the business seemed to be helpful resources but to others they could not understand why we were there*" [O:VS-I:GH]. As previously observed, a lack of support, particularly regarding management, is likely to resort in test engineers losing motivation (see Sections 5.6.4.2 and 6.3.5.3). The interviews also made it clear that the interactors outside of engineering typically wield greater power, which directly influences the amount of V&V applied to a project (see Section 6.3.6.5). Specifically, these interactors use their elevated position to reduce the V&V being performed, usually resulting in the test-based activities being reduced in scope (thus supporting their being seen as dispensable, see Sections 6.3.6.4 and 6.3.6.5, as well as [Rodrigues et al., 2010]), which would, as above, result in the software test activities being prioritised over the security test-based activities. If the time allocated for testing is reduced, there will be a corresponding reduction in scope [Gilbert, 2011]; the interviews showed this results in a focus on primary functionality and a preference to prioritise the software V&V activities to the detriment of security V&V.

Having identified the relative focus and priority which is placed on the activities - by developers, test engineers and organisations as a whole - the interviews helped identify the primary contributor for this. Namely, security V&V encompasses a set of complex activities requiring specialist knowledge. This is, in turn, compounded by a lack of training afforded by the organisations (see Sections 5.6.7.3 and 6.3.8.6). This clearly has implications, most obviously impacting an organisation's ability to improve

their existing level of security V&V practice (which, as found within both phases of data collection, is somewhat limited). Further, we attribute this lack of knowledge to the increase in security V&V outsourcing (see Section 6.3.8.7). As we found, the use of outsourcing is not without problems. Additionally, and more generally, the limited resources of SMEs negatively impacts the V&V activities, in particular, the security V&V activities. Whilst the interviews confirmed that only a subset of an organisation's interactors are actively involved with such activities, they also showed that a number of such "key" interactors are often impacted in finding vulnerabilities by having to fulfil their other responsibilities (e.g. the Architect within one small organisation [O:CT] and the Security Test Engineer within a medium-sized organisation [O:RV]). This echoes the impact of employees, within SMEs, wearing multiple hats [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018] and shows its impact on a specific operational context which is, of course, further compounded by the lack of security V&V knowledge found more generally within an organisation.

In summary, and based on the analysis performed, the interviews support the findings from the first phase, namely, that the software V&V activities have reached a higher level of maturity. As before, this is evident regarding their application, the level of understanding and awareness surrounding the activities themselves, as well as the level of support offered within the organisations e.g. in terms of tool and/or process investment, as well as training in any form. Therefore, it logically follows that the security V&V activities are less likely to involve interactors (which reduces development opportunities e.g. on-the-job learning as "[l]earning is working; working is learning" [Sambrook, 2005, p. 106]) and are more likely to suffer from a variety of undesired interactions. Notably, whilst some of the organisations acknowledged that their security V&V must be improved (e.g. "*[w]e have had some incidents lately with customers that are starting to support my case*" [O:WK-I:RR]), our findings go some way towards explaining the continued rise in the number of vulnerabilities being reported more generally (see Section 1.3.3).

6.4.2 Information Security Culture and V&V Practice

Within the first phase we determined that a relationship exists between an organisation possessing an information security culture and their level of security V&V practice. Specifically, organisations with a stronger information security culture had a more mature security V&V practice (see Section 5.8.2). We continued to explore this theme during the second phase. Additionally, we wanted to establish whether security had become a natural aspect to an organisation's employees' daily activities [Sánchez et al., 2010], whether further improving an organisation's information security culture would

improve their security V&V practice, as well as which would be easier to improve within their organisation.

The interviews strongly suggest that an organisation's information security culture impacts security V&V practice within both small and medium-sized organisations e.g. [O:SS] and [O:RW], with one Product Development Manager [O:SS-I:RD] indicating that security has become "*part of the day-to-day activities, it's part of what they do on a daily basis, as I say, we try to ingrain that from top to bottom, so it's a consideration for everybody, whether they're the MD or whether you're, kind of, a junior customer service representative, in terms of having a, maybe an appreciation, not necessarily doing the day-to-day tasks, but an appreciation of the requirement of our quality software delivery*". However, it is also evident that more could be achieved regarding both improving an organisation's information security culture and security V&V practice. For example, within one small organisation, the Senior Test Engineer [O:DF-I:JR] indicated that their organisation's information security culture has "*had an impact to an extent, but we do need to focus on it more*", elaborating: "*[w]e are all aware and have general security of our activities in place [. . .] but more focus is probably required on our product*". Similarly, the Product Development Director [O:SP-I:WS], within a medium-sized organisation, stated in terms of security becoming a natural aspect of an organisation's employees' daily activities: "*I think the company is very much wanting to be seen as such, but perhaps still needs to improve certain aspects to make it become a fact*". This reflects, more generally, the held "*I think there's always room for improvement*" [O:RW-I:RL] view when it comes to security-focused activities within SMEs.

Further, and although an information security culture should encompass all employees within an organisation, the Security Test Engineer [O:RV-I:BM], within a medium-sized organisation, felt that in the case of "*[e]very employee, maybe not, but definitely it's up there, it's been more, you know, embedded with everything [. . .] it's definitely going that way*". The Security Test Engineer's line manager (a Lead Software Test Engineer [O:RV-I:ML]) also echoed the "*[n]ot yet, no*" view, indicating: "*I think a lot of the training [the Security Test Engineer] will be doing over the next year or two will move us much more towards that*". Notably, this training will encompass the entire organisation: "*working with people like marketing, people like sales and support people, customer-facing roles as well, because the kind of stuff they are going to be pondering is very different to the stuff the engineering team might be pondering*". We recall the importance of involving, and educating, the interactors sitting within the tertiary interactor group (see Section 6.3.3), since they wield considerable influence over the security V&V activities (see, for example, Section 6.3.6.5). The importance of improv-

ing the level of awareness and appreciation for security throughout an organisation is effectively summarised by the Head of Engineering [O:KD-I:MF], based within another medium-sized organisation: "*[w]e need to make the rest of the business understand why it is important so that it is given more priority on the development backlog*". However, as the Lead Software Test Engineer continues: "*as people become more aware of stuff, the trend would be to take more of an active role in it. So that we're kind of aware of just how bad things can be*" (as found, people do not appear ignorant to the impact of security issues see, for example, Section 6.4.1.1). Ultimately, their hope is "*[t]o take a few more steps to practically being able to prevent any problems*" (certainly, improving information security awareness allows employees to move from being unconsciously incompetent to consciously incompetent [Thomson and von Solms, 2006]).

This optimism, that improving an organisation's information security culture would result in an improvement to their security V&V practice, was also observed when interviewing others. For example, in terms of whether their information security culture has pervaded the entire organisation, the Head of Engineering [O:KD-I:MF], within a medium-sized organisation, stated "*[n]ot fully, but it is getting there*". As they noted: "*[w]e have ISO27001 certification, and that has ramifications throughout the business*". This example also supports the observed impact of external influences on security V&V (see Section 6.3.5.4), as well as through customer demand [McGraw, 2006, Asghari et al., 2016, Owsley, 2017] (the Security Test Engineer [O:RV-I:BM] also emphasised the impact of external influences, see Section 6.4.1.1).

Whilst there was some variation in the reported extent that an organisation's security V&V practice could be improved by improving their information security culture, the interviews reinforced that improving an organisation's information security culture would bring an increased level of awareness to the security V&V activities themselves (we recall that "[a]wareness is the cornerstone of a security culture" [Williams, 2009, p. 52]). Although this was strongly asserted by several interviewees (e.g. as indicated by the Product Development Manager [O:SS-I:RD] within a small organisation and the Product Development Director [O:SP-I:WS] within a medium-sized organisation), others were a little more cautious in attributing a direct relationship. For example, one Test Manager [O:RW-I:RL] vocalised that "*I think it would definitely have a contribution towards it*". Exploring this "contribution" further, it was felt that only those within engineering could improve security V&V practice from a technical perspective i.e. with those outside being more concerned with the organisation's information security culture and only being able to provide support, and possibly some level of direction e.g. "*they would give you the talk on, you know, what best practice is*". As they subsequently acknowledged: "*I don't think that kind of knowledge would kind of necessarily*

help people, it makes them aware of it, but they don't know what to do with it". Therefore, improving security V&V practice "*would have to be driven independently*" but would, in turn, benefit from the increased level of awareness obtained from improving the information security culture more generally within the organisation.

The view that information security culture and security V&V practice are two distinct aspects was echoed by others e.g. the Senior Test Engineer [O:DF-I:JR], within one small organisation, indicated that "*the security aspects with regards to the product is different to the information security culture*" and the Head of Engineering [O:KD-I:MF], within a medium-sized organisation, indicated that they did not believe further improving their organisation's information security culture would help improve their security V&V practice, observing: "*our practice is already good, it is detailed knowledge of security vulnerabilities that we are lacking*". This echoes the already observed impact of a lack of security V&V-related knowledge (see Section 6.3.8.6). Similarly, the Security Test Engineer [O:RV-I:BM], within another medium-sized organisation, felt that "*they're quite separate*", however, they vocalised that "*from me and other people pushing security and trying to, you know, push everything about security onto all different parts of the company, more people are aware of it, more people know different things, more people have to do things, because we implement, you know, rules on, say, software you can use and stuff*". Ultimately, this results in "*more awareness*" within the organisation as a whole. Therefore, the interviews confirmed the importance of cultivating such an awareness since, according to the Software Tester [O:CT-I:DS] within another small organisation (this one without an information security culture), improving their information security culture would help improve their security V&V practice: "*if you are talking about the entire organisation, from top to bottom; clearly, if the CEO and the board were more aware of the bad things that could happen . . . its not necessarily they're not aware of it all, it's just its so far to the back of their mind people have almost forgotten about it. You know, these are business people, you know, they're not quite working on the same level as the likes of us. So, yeah, if there was an improvement in [information security culture] through the whole organisation then they would then be saying what have you done about security testing . . . they would be asking questions of us, you know, have you done this, or what are you going to do about this*". Whilst this is reflective of an organisation sitting at the early stages in the Information Security Competence Maturity Model i.e. with employees showing both an unconscious incompetence and conscious incompetence when it comes to information security [Thomson and von Solms, 2006], it also shows the importance of senior management being actively involved with their organisation's information security (see Section 5.7.3) and security V&V (see Section 6.3.5.3).

That there is a relationship between an organisation's information security culture, and their level of security V&V practice, as well as that their respective levels of maturity can vary, we find opinion divided on which is easier to improve. For example, the Test Manager [O:RW-I:RL], within one medium-sized organisation, stated that improving their organisation's information security culture would be easier. Exploring this, we find continuation in one of the identified underlying themes impacting security V&V practice more generally, namely, that it encompasses a set of complex activities (see Section 6.3.6.6). As summarised by the Test Manager: "*[v]erification and validation it's pretty much harder to know how to understand how to do that*". Similarly, but within a small organisation, a Senior Test Engineer [O:DF-I:JR] felt that their organisation's information security culture would be easier to improve since "*I do not think we have all the knowledge at an engineering level to implement the required changes for V&V practice*". However, it was also acknowledged that they "*have an established product in some respects and making changes to encourage a fully compliant security is more than the resource we have available*", which echoes the impact an SME's restricted resources can have on security V&V (see Section 6.3.8 regarding the impact of resource flows).

In contrast to the above, the Product Development Director [O:SP-I:WS], within a medium-sized organisation, felt it would be easier to improve security V&V practice since "*there is too little real understanding of what is needed at the higher levels*". In conjunction, there is "*a lot of understanding and will within the engineering teams to improve things*". However, without senior management support there is a danger that practice will become inconsistent, since it became evident that "*there is more than one engineering team*", leading to the view "*so perhaps each will take its own path*". This highlights a potential problem when a common process does not exist (we recall that the security V&V activities are less likely to have any form of governing process, see Section 5.6.3.2). It also supports the view that within "immature software development organizations, the processes used differ across projects because they are based on the experiences and preferences of the individuals assigned to each project, rather than on common organizational practice" [National Research Council, 1996, p. 30]. The Product Development Manager [O:SS-I:RD], within a small organisation, also vocalised that security V&V practice would be easier to improve since "*it's just a task and easily measurable*". However, as they continued: "*I think but the more relevant one [to improve] would be the culture, because then that will ensure as a by-product of the culture you're doing the actual tasks anyway*". An example which further supports the relationship between an organisation's information security culture and their security V&V practice.

In summary, the interviews confirmed many of the findings from the first phase.

Specifically, showing that organisations with a stronger information security culture typically have a more mature security V&V practice. For instance, within the organisation [O:RV] with the dedicated security V&V resource, we found, when asking a Lead Software Test Engineer [O:RV-I:SK] whether there was an information security policy within the organisation, the answer was immediate: "*yes, there is . . . that part is definitely in our terms and conditions in our contract*" (this supports the finding from the first phase, in that organisations enforcing their information security policy within their employees' contract of employment are more likely to report a healthier information security culture, see Section 5.7.3, which further supports the need for organisations to enforce their information security policies [von Solms and von Solms, 2004, von Solms and von Solms, 2009]). However, this organisation is clearly an exemplar when it comes to establishing an information security culture e.g. there were references to "*company confidentiality*" in terms of the software currently undergoing development, for "*any communications that we have internally are made very clear that it's all confidential*" and "*we all lock our computers*". In terms of the latter, it was indicated, in jest, that this was also driven by a "*just because what other engineers are like on occasion*" view i.e. "*unless you want someone to switch your mouse buttons around*" which, although a light-hearted example, shows that engineers, within this organisation, have developed a security-focused mindset due to the existence of the wider organisation's information security culture. It was also indicated that "*we have frequently asked questions on our website*" to help people understand the value of security. On enquiring whether we stood outside of the room within their organisation where the interview was being conducted, and posed a set of these questions to ten employees at random, would they be able to answer appropriately, the response conveyed a confidence, reflective of an organisation with a more mature information security culture e.g. "*the sales guys would definitely be able to . . . talk that up as you would expect sales people to be able to*" and "*nobody in engineering would ever side step a security issue*". This also helps show that the information security culture has permeated throughout the organisation.

Further, it is apparent from the interviews conducted, that any improvements to security V&V practice will have to be driven by engineering (which is where the primary interactors are situated, see Section 6.3.1), however, they will require support from the wider organisation. For example, although one Head of Engineering [O:KD-I:MF] acknowledged that their organisation's information security culture was driven by ISO27001 certification (leading to security becoming a more natural aspect to an employees' daily activities [Sánchez et al., 2010]), they also indicated "*[w]e have performed security testing on our applications for many years, long before we went for ISO27001 accreditation*". This sentiment was echoed by others e.g. a Product Devel-

opment Director [O:SP-I:WS] indicated: "*[s]o far, the vast majority of security V&V has been put in place due to the engineering team knowing it is the right thing to do*" i.e. regardless of the wider organisation's information security culture. However, it is clear that as information security culture raises the level of awareness of security more generally within an organisation e.g. from "*top to bottom*" (as echoed by several interviewees e.g. [O:SS-I:RD] and [O:CT-I:DS]), this will help raise the profile and necessity of security V&V to those with the power to support the activities (i.e. the tertiary interactors, see Section 6.3.3 - who can clearly influence the security V&V activities, see Sections 6.3.6.4, 6.3.6.5 and 6.3.8). In the words of the Head of Engineering: "*[w]e need to make the rest of the business understand why it is important so that it is given more priority on the development backlog*". We find that this can be achieved by improving an organisation's information security culture, which further supports the view that an information security culture can, and does, influence security V&V within UK-based software SMEs.

6.4.3 Reflections

Whilst the interviews afforded us greater opportunity to explore responses (both those obtained from the first phase and when conducting the interviews), their interactive nature also enabled a rapport to develop with the interviewees, in turn, improving the quality of the responses received [Singer et al., 2008]. This was important, since: "[w]ithout rapport, even the best-phrased questions can fall flat and elicit brief, uninformative answers" [Leech, 2002, p. 665]. It became clear, throughout the interviews, that a level of rapport had developed with the interviewees e.g. within one small organisation the Software Tester [O:CT-I:DS], on referring to the "*business people*" within their organisation and their perception of security, stated: "*they're not quite working on the same level as the likes of us*" and, within a medium-sized organisation, the Scrum Master and Lead QA Engineer [O:VS-I:GH] indicated: "*[s]ome of these questions are awesome and should be asked on a daily basis!*". Generally, the interviews were conversational in tone, which we attribute to the shared level of understanding which existed between interviewer and interviewee. Whilst some studies highlight the importance of shared knowledge and identity when interviewing [Shah, 2004], others indicate that it is easier to study areas when the researcher has had little exposure [Johnson and Weller, 2001]. However, we, like [Hove and Anda, 2005], believe that when researching areas of software engineering, it is vital for the interviewer to possess extensive knowledge of the area being explored. The inherent complexity of security V&V (see Sections 1.3.3 and 1.3.5) emphasises this.

This rapport led the interviewees to feel sufficiently comfortable to reflect honestly

upon both their organisation's - and their individual - strengths and weaknesses. Their honesty, when reflecting upon their own competence (e.g. one Lead Software Test Engineer [O:RV-I:SK], when discussing code reviews, indicated: "*C++, I wouldn't be able to read my way through the code . . . I personally stay away from that*" and, a Software Tester [O:CT-I:DS], confessed they were "*not capable of doing*" the necessary updates to their organisation's test framework), are indicative of an open and honest discussion. Similarly, interviewees shied away from painting a positive or unrealistic picture of their organisation e.g. indicating "*at the minute, security stuff would be a bit more reactive*" [O:RV-I:ML] and that some tasks have been "*hacked together to get things to work and it's a mess*" [O:CT-I:DS]. Thus, although interviewees were aware that security was the focus of the research, they still felt able to articulate "*it is not that there is no security [. . .] there is, you know, such a concept of security in the product, but it's not the number one priority*" [O:CT-I:DS]. Whilst this improves the level of confidence we have in the quality and integrity of the data obtained, further support is afforded when observing that many of the interviewees were also involved in the first phase (see Appendix L). Specifically, we found, when comparing the responses of an individual - across both phases of data collection - that no individual appeared to contradict themselves e.g. with one Test Manager [O:RW-I:RL] indicating: "*I wouldn't say since I last answered that question a significant amount has changed on the security side*" (which, in turn, heightens our confidence in the data obtained during the first phase).

In conclusion, the second phase provided additional insight into how security V&V is practiced within UK-based software SMEs. In particular, it was able to corroborate many of the findings from the first phase. Specifically, helping confirm the relative maturity of software and security V&V in terms of their application, as well as the associated levels of understanding, awareness, and support afforded. The interviews also supported the view that organisations with a stronger information security culture would typically have a more mature security V&V practice - but one which is still less mature than their software V&V practice. Additionally, the interviews helped convey that improving an organisation's information security culture would lead to improvements in their security V&V practice.

Having concluded the second phase of data collection and analysis, it is within Chapter 7 that we examine (i.e. mix) the two phases in order to address the research objectives and to position our findings in context of the existing literature.

Chapter 7

Discussion

Within this chapter we discuss, and bring together, the findings obtained during the two phases of data collection and analysis performed. Specifically, this chapter provides:

Section 7.1: We reintroduce the thesis research objectives and the use of STIN for studying security V&V within software SMEs.

Section 7.2: We synthesise the two phases of data collection and analysis conducted, using STIN to structure our analysis.

Section 7.3: We present a series of discussion themes which underly the synthesised analysis.

Section 7.4: We conclude our discussion by re-examining the specific thesis research objectives and reflecting on the devised conceptual model.

We begin by providing an introduction to the ensuing discussion.

7.1 An Introduction

Within this thesis we sought to understand how UK-based software SMEs practice security V&V and how, as a process, it is influenced by the socio-technical characteristics of such organisations. Aside from examining a critical but, rather disturbingly, underexplored area of software engineering practice (see Section 1.3.6 and [Kreeger and Harindranath, 2012]), we also sought to establish whether organisations could possess a mature security V&V practice without a correspondingly mature software V&V practice. Additionally, we aimed to determine whether an organisation's wider information security culture influenced security V&V practice.

Therefore, and given the dearth of existing knowledge, we employed a mixed methods approach to address the overarching research question, and the associated research objectives, structuring the phases of data collection sequentially (see Section 4.2). Whilst the first phase employed a survey, the second phase involved a series of semi-structured interviews. This enabled several themes uncovered during the first phase to be more deeply explored and helped corroborate many of the findings from that phase (thereby echoing the utility within [Greene et al., 1989, Bryman, 2006, Zheng et al., 2017]). Further, and acknowledging the socio-technical nature of V&V (see Section 3.3.6), it was essential to adopt a socio-technical approach when framing the study. We elected to adopt STIN due to its providing a structured, but flexible framework, capable of being applied to a variety of complex technologically and non-technologically mediated social settings. In particular, [Scacchi, 2005, Amrit, 2008, Jensen, 2010] found it well-suited for studying aspects of software engineering. It was by adopting STIN that we were able to create a conceptual model (see Figure 3.2). This not only helped reflect its suitability for studying V&V, it also helped structure and frame our analysis.

Having now completed two phases of data collection and analysis (Chapters 5 and 6), we begin by synthesising the data obtained.

7.2 Synthesising Two Phases of Data Collection

By adopting a mixed methods approach, we acknowledge the need to synthesise the data obtained. This is necessary to show what has been learned overall in terms of the identified research objectives [Creswell and Plano Clark, 2011]. As before, we structure our discussion by first revisiting the interactors and their incentives and then the relationships which exist within the V&V socio-technical network.

7.2.1 Interactor Analysis Synthesis

The V&V socio-technical network, as found within UK-based software SMEs, contains a rich, diverse set of interactors - both human and non-human, technical and non-technical - with each interacting with one another, both internal and external, to an organisation. This emphasises the collaborative nature of software engineering and V&V [Aurum et al., 2002, Burnstein, 2003, Tervonen and Harjumaa, 2004, Yilmaz and Phillips, 2006, Hass, 2008, Martin et al., 2008, Whitehead et al., 2010, DeFranco and Laplante, 2017]. However, this richness in diversity is much more apparent in terms of the software V&V activities than the security V&V activities, thereby helping support the view that software SMEs prioritise other quality attributes over security. This does not reflect the acknowledged importance of security (see Sections 1.3.4 and 1.3.6).

Interactor diversity aside, developers and test engineers are the primary interactors across both sets of activities. Whilst this by itself is unsurprising, it helps identify where focus should be on improving V&V within an organisation - specifically, from a technical perspective (e.g. the application of the activities) - since, ultimately, these are the interactors best placed to address them (both in terms of their assigned roles and responsibilities, but also through their collective experiences e.g. their educational backgrounds). However, there are two findings - each supported by both phases of data collection - which show that whilst these two groups of interactors are hierarchically situated and positioned within the same function (i.e. engineering), within an organisation, they are clearly distinct from one another (see Sections 5.6.1 and 6.3.1). Firstly, all organisations actively, and formally, distinguish between the two types of interactor which, given the wide of adoption of Agile, was surprising as this usually advocates the engineering generalist and the blurring of roles [Meszaros and Aston, 2007, Crispin and Gregory, 2009]. Secondly, we find the traditional role divide, in that developers are more likely to be involved in review-based activities and test engineers in test-based activities, continues to be perpetuated throughout the organisations. In some instances this was expected, however, with the majority of the organisations proclaiming to adopt Agile (see Figure 5.4), we lean towards the view that many of these organisations may be in a transitory phase in their adoption (a view borne out, to some extent, by the fact that many of these organisations also adopt other software development methodologies as well e.g. Waterfall and the V-Model). However, we should not overlook the general upheaval and complexity associated with organisations transitioning in how they work which, without an appropriate level of training, may lead to an incompleteness in terms of Agile understanding and its adoption, collectively, resulting in an Agile veneer (as found by [Denning, 2018]). We discuss the persistence of traditional working practices, in the face of Agile adoption, within Section 7.3.1 and show that its adoption can

negatively impact some security-related activities.

Although this distinction, between the primary interactors, is not necessarily negative in itself (various maturity models consider an independent test group a sign of maturity e.g. [TMMi Foundation, 2018]), the divide between these two interactors is likely to exacerbate conflict (as evidenced within Sections 6.3.6.2 and 6.3.6.3). Ultimately, we believe that this role divide - and thus the continued distinction between the two types of interactor - is driven by each possessing different skill sets, mindsets, mental processes and personality attributes [Pettichord, 2000, Cohen et al., 2004, Dhaliwal et al., 2011, Zhang et al., 2014, Zhang et al., 2018], as well as by the deeply entrenched view held by developers (and, as we found, by other interactors within an organisation, see Section 6.3.6.4) that test-based activities are second-class or dispensable [Bertolino, 2003, Hutcheson, 2003, Rodrigues et al., 2010]. Notably, a view whose beginnings are evidently cultivated within the classroom, with test-based activities typically underemphasised [Murrill, 1998, Jones, 2000, Jones and Chatmon, 2001, Leska, 2004, Scott et al., 2004, Kazemian and Howles, 2005, Elbaum et al., 2007, Kreeger, 2009, Bajaj and Balram, 2009] (the impact of this was borne out by the Security Test Engineer [O:RV-I:BM] not initially knowing what a test role would encompass, see Section 6.3.1). Therefore, rather than striving to blur the roles, organisations should seek to reduce the barriers between them. In particular, they should actively instill a level of test awareness, support and responsibility throughout e.g. by ensuring that the time assigned to a project's test-based activities is not the first aspect reduced when faced with a deadline (as it usually is [Murugesan, 1994, Bashir and Goel, 1999, Nguyen et al., 2006, Desikan and Ramesh, 2006], see Section 6.3.8.8), as well as ensuring that a developer's job description, and their responsibilities, includes testing (the necessity to enforce compliance has been found more generally regarding an organisation's information security practices [von Solms and von Solms, 2004, von Solms and von Solms, 2009] and, whilst software engineers are required to follow processes [O'Regan, 2011], they usually need to be enforced [Testa, 2009] - thereby implying a level of management engagement and support - which is something we did not find, see Sections 5.6.4.2 and 6.3.5.3). Further, and acknowledging the importance of communication, any physical barriers (e.g. developers and test engineers sitting in different parts of an office) should be removed i.e. there should be regular communication, regardless of mechanism, between the two interactors. Actively improving the communication between these interactors would help reduce any potential frustrations and resulting conflict [Sawyer, 2001, Zhang et al., 2014].

Further, and although it was genuinely exciting to find a dedicated security V&V resource within one of the SMEs (see Section 6.3.1), the lack of interactors - situated

within the organisations - with any official responsibility for performing a security-focused V&V activity (e.g. a developer recognised for performing security code reviews or a test engineer who also performs penetration testing) was surprising. This was especially so given the types of products being developed by the organisations - coupled with their seeming level of acknowledgment of how important security is regarding business reputation, retaining customers and entering new markets. Certainly, and given the contexts in which some of these organisations' products were ultimately being deployed, we find the lack of internal expertise quite disturbing. Further, this lack of dedicated, capable resource has resulted in a reliance on the outsourcing of these activities to the apparent detriment to an organisation's own internal development and growth. This is discussed further within Section 7.3.3, however, it became abundantly clear, particularly within the interviews, that security V&V is complex [Austin and Williams, 2011] and encompasses activities that, at best, sat on the periphery of an employee's level of understanding and awareness. Ultimately, this helped generate a 'someone else's problem' view (this is discussed further within Section 7.3.2).

Generally, this lack of awareness, and the associated support afforded to the security V&V activities within an organisation, was also reflected in the non-human interactors - in particular, the tools and processes necessary for performing the activities. This was exhibited, across both phases of data collection (see Sections 5.6.3, 6.3.4 and 6.3.5.5), by the lack of such interactors, as well as a respondent's and interviewee's general inability to discuss them (again, an area exacerbated by the reliance on security V&V outsourcing, see Sections 5.6.2 and 6.3.8.7). Ultimately, the findings concerning the non-human interactors - which are essential to performing effective security V&V (especially given an SMEs inherent resourcing constraints e.g. through automating incredibly complex tasks) - show that security V&V has not reached the same level of maturity, and support, as software V&V within UK-based software SMEs. This was generally the case regardless of organisational size, age and product focus (we recall the diversity of the organisations studied, see Section 5.4.1).

The reduced diversity of interactors involved with the security V&V activities also impacts interactor motivation - both intrinsic and extrinsic - in terms of performing the associated activities. For example, without an appropriate set of processes, and without the necessary supporting tools, interactors are less likely to receive either the direction, or the support necessary, to successfully perform the activities e.g. through using tools to automate security code reviews (such a task has been considered cumbersome [Edmundson et al., 2013], therefore, it is clearly important to reduce the burdensome nature of the task, through automated tools [McGraw, 2008], to further avoid demotivating interactors). However, we find that the interactors are primarily

motivated to perform both sets of V&V activities in order to improve software quality (see Sections 5.6.4.1 and 6.3.5.1). This supports the existing literature on motivation within a software engineering context e.g. [Beecham et al., 2008, Hall et al., 2008], however, we confirm that such intrinsically-driven motivation is also exhibited in terms of security V&V within UK-based software SMEs. Given that interactors are more likely to be motivated in this way in terms of the software V&V activities, we initially attributed this to interactors not affording importance to security as a quality attribute. However, and demonstrating the importance of performing more than one phase of data collection, as well as the value of adopting a mixed methods approach, we found evidence refuting the view that interactors were simply ignorant to the impact of security failings (see Section 6.4.1.1). Therefore, although we can infer that there is a general level of awareness of the implications of security failings within software SMEs, this awareness is distinct to that which surrounds the security V&V activities themselves (where there is a general lack of awareness amongst both technical and non-technical stakeholders, see Section 7.3.2). Notably, where there were stronger levels of awareness (i.e. within organisations with a more firmly established information security culture) we found a more mature security V&V practice (see Sections 5.8.2 and 6.4.2). This helps confirm that awareness is the 'cornerstone' of an information security culture within small organisations [Williams, 2009]. It is also apparent that security, and security V&V awareness levels - and the associated motivation (in this case, extrinsic) - are cultivated within SMEs through a number of external factors e.g. standards, regulations and certification (we discuss this further within Section 7.3.4).

In summary, although there is evidently recognition, and a degree of awareness associated with performing the security V&V activities, we find that they are impacted by several of the relationships which exist within the V&V socio-technical network. Ultimately, this results in the security V&V activities being performed less frequently, less efficiently and with less confidence than the software V&V activities.

7.2.2 Network Relationship Analysis Synthesis

In conjunction with the diverse set of interactors, and their incentives, we find that their relationships with one another are equally diverse and rich. Identifying these relationships not only helped identify how these interactors interact, but it also enabled us to understand the context in how they view, and influence, one another - both positively and negatively. Notably, the relationships within the V&V socio-technical network are generally more inclusive, and supported, in terms of the software V&V activities, thereby implying that the security V&V activities are less inclusive, and are less supported, within UK-based software SMEs.

In particular, although communication between the interactors was generally assisted by their close proximity to one another (especially regarding the primary interactors, see Section 6.3.7.1), as well as through the use of a common set of shared communication mechanisms and tools - across all the V&V activities - we find that the general lack of security expertise possessed by the interactors limits communication in terms of the security V&V activities. This reflects its inherent complexity [Kreeger, 2009, Wang and Guo, 2009, Austin and Williams, 2011], with it being unrealistic to expect interactors, both technical and non-technical, to discuss and engage with a subject when they have a relatively little understanding of the topic. It also emphasises the importance of considering expertise as a resource flow (as it is within STIN [Meyer, 2007, Taylor-Smith, 2016]). Specifically, interactor communication appears to be hindered by a lack of understanding and, to some degree, a lack of awareness (which we find is often compounded by a lack of interactor inclusivity, for example, with only a subset of an organisation's engineers communicating with the engaged outsourced engineers on which the organisation relies for their security-focused testing, see Sections 5.6.6.1 and 6.3.7.4). A lack of understanding and awareness (and the associated exclusivity) appear less of a concern for the software V&V activities, which perhaps should be expected given the greater shared understanding - and language - existing between the interactors in terms of these activities. Therefore, with regards to both sets of activities, we find that interactor communication is not limited by opportunity (regular meetings and stand-ups were common place, see Section 6.3.7.2), or the available communication mechanisms and associated tools (interactors appeared comfortable with both their technologically and non-technologically mediated social relationships and were found to utilise a common, and accessible, set of tools within their organisations in which to communicate e.g. centralised defect tracking and version control systems, see Sections 5.6.6.3 and 6.3.4.1).

However, although our findings concerning communication were generally positive, we should not overlook the social dynamics which exist between the two primary interactors (i.e. developers and test engineers), since it is here where we find that communication can be hindered by their differing capabilities, objectives, personalities and mindsets [Sawyer, 2001, Cohen et al., 2004, Zhang et al., 2014, Zhang et al., 2018]. Whilst we do not believe that their relationship can be improved solely through an improvement in the frequency and type of communication (and we find evidence to suggest that it does need to improve, see Section 6.3.6.3), it is likely, with an improvement in the adoption of a more transparent - and inclusive - software development methodology, such as Agile, that we will continue to see improvements in the relationship between the primary interactors (although this demands a level of adoption

exceeding an Agile veneer [Denning, 2018]). Improving the relationship between the two primary interactors, within the V&V socio-technical network, is clearly important. Conflict, especially surrounding the communication of defects and vulnerabilities, can be lessened when more information is communicated by both parties (e.g. from how to reproduce an issue, to how an issue was addressed). Although, in terms of vulnerabilities, we find that such communication is compounded by a lack of, or insufficient, interactor communication - which is typically due to the interactors involved only possessing a general understanding of security [McGraw, 2004, Wurster and van Oorschot, 2008, Kreeger, 2009, Austin and Williams, 2011]. This can further impact interactor communication, with the added complexity of security potentially exacerbating conflict. However, more generally, we would have expected the adoption of Agile to have improved the relationship between the two interactors - in particular, helping erode the distinction in their roles (we discuss this further within Section 7.3.1). Therefore, whilst the traditional divide between the two interactors appears to continue, perpetuating the view that test engineers are second-class individuals, performing second-class activities [Bertolino, 2003, Juristo et al., 2006] (see Sections 6.3.6.2 and 6.3.6.3), we find that such views are also held by other interactors within the organisations (see Section 6.3.6.4). This is particularly damaging to the security V&V activities where, in some cases, it is a view harboured by the interactors who control the resource flows. We also find that these interactors perceive some of the activities dispensable (test-related activities in particular, see [Rodrigues et al., 2010] and Section 5.6.5.2) and encourage focus on a product's primary functionality. Given that these interactors typically wield greater power within the V&V socio-technical network (see Section 6.3.6.5), further emphasises the need to focus on improving security V&V communication, and the raising of awareness levels, throughout an organisation. We recall the importance of an organisation's management being actively seen to support V&V activities (for example, any reduction in test time can lead to the activities being perceived as having little value [Cohen et al., 2004], with a lack of management support, more generally, known to negatively impact test engineer motivation, causing them to lose interest in their tasks [Perry, 2006]), however, we also recall our findings which show that the interactors involved with V&V were not typically motivated by their organisation's management in terms of performing any V&V activity (see Sections 5.6.4.2 and 6.3.5.3).

In addition, we highlight that improving communication more generally, between all interactors within the V&V socio-technical network - both those internal and external to an organisation - would further help address the exclusivity associated with the security V&V activities. For example, we find a need to improve the levels of communication with customers and outsourced engineers (both of which are influential in terms

of security V&V, see Section 7.3.4). Ultimately, a more transparent, and inclusive, approach to communication would assist both sets of V&V activities, however, it is more likely to have greater impact in terms of security V&V, especially given the reliance on outsourced engineers for the associated activities (see Sections 5.6.2, 6.3.8.7 and 7.3.3) and with customers generally becoming more security-focused and technically astute [Zabicki and Ellis, 2017]. Such communication would also increase the levels of awareness which surround the security V&V activities - notably, throughout an organisation - and, therefore, help address the topic of interactor inclusivity in terms of these activities. It would also give greater prominence to the activities within an organisation, helping counter the observed lack of management involvement. However, to be successful in these aims, there would need to be a noticeable improvement in the level of security V&V knowledge within an organisation. Unfortunately, we found training opportunities limited in terms of opportunity and frequency (see Section 5.6.7.3). This has a more significant impact on the security V&V activities (see Sections 5.6.5.1 and 6.3.8.6).

It is also apparent that organisations should strive to establish a consistency of practice amongst their employees (many maturity models consider this a sign of a more mature organisation e.g. [CMMI Product Team, 2010]). A key example of this concerns the communication surrounding vulnerabilities i.e. the most fundamental object of communication in a security V&V context. Specifically, we find that many organisations do not explicitly identify security-related issues within their adopted defect tracking systems (which should be the single source for information on which the surrounding communication, regardless of mechanism, is based, see Section 6.3.7.3). This calls to mind the expression 'hunting for a needle in a haystack' (since unlabelled vulnerabilities can easily become lost amongst defects [Peters et al., 2017]). Whilst some organisations label security issues, improving their ability to communicate this information internally and externally e.g. to customers (a practice that should be adopted by organisations [Russell and Van Duren, 2016]), we find that such practice is inconsistent at best. Ultimately, this reduces the ability to understand the scope of the security problem within a product - which is not a good starting point given the evident complexity of security V&V and security engineering more generally. This decreases the potential for identifying vulnerable areas of code (e.g. [Neuhaus et al., 2007]) which, in turn, reduces the ability for organisations to focus their security V&V efforts. As we found, SMEs are constrained in their human and non-human resources, expertise and training opportunities (see Sections 5.6.7 and 6.3.8). Whilst this is reflected within many existing studies e.g. [Habra et al., 1999, Ward et al., 2001, Andersson and Runeson, 2002, Murthy, 2009, Tosun et al., 2009, Basri and O'Connor, 2010, Cruz-Cunha,

2010, Woschke et al., 2017], we identify its impact on a specific organisational and technological context, namely, that of security V&V within UK-based software SMEs. Such organisations should seek all opportunities to best capitalise on the security information - and expertise - they have available.

Ultimately, the nature of security V&V, and its inherent complexity, reduces the ability for interactors - without an appropriate level of understanding - to converge around, and discuss, the associated activities. This reduces the ability for many interactors to participate, in turn, leading to a greater deal of exclusivity in terms of the security V&V activities. This reinforces the 'someone else's problem' attitude, leading to a reduced level of awareness of the issues and the current state of practice within an organisation (which reflects the observed "Do Not Know" trend encountered in terms of security V&V, see Sections 5.5, 5.6.3.1 and 5.6.3.2 - we discuss the issue of awareness further within Section 7.3.2). Unfortunately, many of the problems inherent within SMEs (e.g. resourcing constraints [Sitnikova et al., 2007, Cruz-Cunha, 2010]) and software SMEs (e.g. competitive pressures [Feldmann and Pizka, 2003]) further compound an already weakened area of an organisation's software development process - with the software V&V activities being less impacted by complexity, therefore, increasing their inclusivity and the associated levels of awareness and support. Thus we can conclude that the software V&V activities sit at a higher level of maturity than the security V&V activities. Notably, this is apparent in terms of their application, associated inclusivity and their surrounding levels of understanding and awareness. In summary, this helps explain why we continue to see a rise in the number of vulnerabilities being reported, as well as the continued repeating of the same security-related mistakes (see Section 1.3.3).

7.3 Discussion Themes

Having performed two phases of data collection (Chapters 5 and 6), we now identify, and discuss, several themes that surfaced during data analysis which, in some instances, crossed several of the key themes within STIN.

7.3.1 Security V&V and Agile Adoption

We found, to varying degrees, that the majority of organisations studied have actively embraced Agile and its associated ceremonies and practices e.g. scrum teams have been formed, backlogs exist and daily stand-ups are common place. This reflects the current popularity of organisations embracing Agile [Clarke and O'Connor, 2013, Ryan and O'Connor, 2013, Rindell et al., 2015, Chóliz et al., 2015, Rindell et al., 2017],

however, despite this, it is apparent that the benefits extolled from such adoption have not yet materialised across all of the organisations and, most notably, it does not appear to have contributed towards a noticeable improvement regarding their security V&V practices. In some instances, this can potentially be attributed to several of the organisations having only recently embraced Agile, thus they are still finding their way and are probably encountering several of the expected transition troubles [Puleio, 2006], with [Nerur et al., 2005] explicitly indicating this for organisations typically versed in traditional software development methodologies (as they elaborate, neither culture, or mindsets, can be easily changed) and [Crispin and Gregory, 2009] indicating similar in terms of organisations with existing, independent test teams. However, many organisations are more established in their adoption, which suggests that undesired interactions and influences are being encountered that are impacting V&V practice which, ultimately, helps support the view that Agile is not a panacea (a view echoed by others e.g. [Cao and Ramesh, 2008, Rigby et al., 2016]).

For example, a key Agile principle is pairing people to work together [Shaye, 2008]. As Agile is a co-operative social process, it is vital that team members value and trust each other [Nerur et al., 2005]. However, we observe that the traditional working practices, and views, of the two primary interactors - namely, developers and test engineers - are still deeply ingrained in, and exhibited by, both types of interactor. This does not conform to the held view that Agile completely redefines quality assurance work [Talby et al., 2006]. Specifically, we find that a confrontational attitude is still present, with a developer's perception of test activities, and test engineers, still very much negative (whilst this reflects the findings of others e.g. [Juristo et al., 2006], many are not positioned within both an SME and an Agile context, however, the latter is observed within [Watkins, 2009]). These attitudes (discussed within Sections 6.3.6.2 and 6.3.6.3) have perpetuated the separation between the interactors in terms of specific V&V activities. Therefore, although everyone should write tests on Agile projects [Talby et al., 2006], we found little evidence that developers actively embrace such tasks. Similarly, we found test engineers were not involved with review-based activities. However, it is acknowledged that all team members should be considered equal, capable of undertaking design, implementation and test-based activities i.e. no strict roles (e.g. test engineers) should exist [Maier et al., 2017]. Such a view is exemplified by [Puleio, 2006] i.e., within Microsoft, developers learnt how to test and test engineers to write production code (notably, this team were, at least initially, permitted to experiment when adopting Agile without facing hard deadlines - this emphasises the importance of management support, but also reflects an operating environment unlikely to be found within SMEs given the resourcing contraints and competitive pressures generally faced

e.g. [Feldmann and Pizka, 2003, Basri and O'Connor, 2010]). It is thus important for people to not assume that their previous roles and responsibilities will remain the same when transitioning to Agile [Sumrell, 2007]. However, even when moving to Agile, managers may continue to reinforce this role divide by associating employees with specific roles [Boehm and Turner, 2005] (as Boehm and Turner observe, this can impact the multi-tasking nature of an Agile team). Therefore, whilst team members should not be confined to a specialised role [Nerur et al., 2005] (Agile requires the full integration of testing and development activities [Talby et al., 2006]), we did not find such levels of integration within the organisations proclaiming to practice Agile - this perpetuates the long established views held by the interactors, impacting both software and security V&V.

Similarly, the continued preservation of existing working practices is also grounded in the observation that many organisations actively embrace multiple software development methodologies (see Figure 5.4). This suggests a reluctance to let go of traditional working practices - which are best exemplified by the Waterfall model and the 'throw over the wall' mindset. However, the use of multiple methodologies, especially a combination of traditional and Agile, has been found in other organisations, leading to the term 'Water-Scrum-Fall' surfacing [West, 2011, Theocharis et al., 2015]. Notably, [Boehm and Turner, 2005] observe the difficultly of working with different methodologies (they provide the example of merging lightweight Agile processes with standard industrial processes in a way without impacting either). Interestingly, [Crispin and Gregory, 2009] state that it is the test engineers who often take the longest time to transition to an Agile environment (with [Talby et al., 2006] also indicating more generally (i.e. regardless of interactor type), that fully adopting quality practices, in the context of Agile, is a slower and more difficult journey). This is echoed by others e.g. [Sumrell, 2007], with [Puleio, 2006] indicating that their team (based within Microsoft) initially thought that testing would be the easiest aspect of the Agile adoption process. Therefore, whilst we found that test-based activities already suffer through a developer reluctance to participate, it is apparent that the difficulties encountered when transitioning to Agile further compound these activities (this has also been observed in the working experience of the author). Notably, test engineers must change their mindset if they are going to successfully become part of an Agile team [Talby et al., 2006].

Further, many established secure development methodologies are considered more suited to the Waterfall and the V-Model, as opposed to Agile-based environments [Maier et al., 2017]. For example, Microsoft's SDL, CLASP, Touchpoints and Common Criteria all predate Agile [Rindell et al., 2017]. Thus, as Rindell et al. state, this can

lead to initial difficulties with their adoption and application within an Agile environment. Baca and Carlsson also found that such methodologies scaled badly in an Agile context (in particular, the design and test phases) [Baca and Carlsson, 2011] (that this observation is positioned within the context of a global organisation, we feel that the inherent resourcing constraints of SMEs would further compound the situation). Therefore, organisations will potentially encounter conflict when desiring to embrace Agile and secure development practices ([Sinnhofer et al., 2015] state that Agile is not used for the development and evaluation of secure products). Notably, although we found that external certifications, such as FIPS 140-2 and Common Criteria, are strong motivators in terms of organisations performing security V&V (see Sections 5.6.4.5 and 6.3.5.4), Common Criteria has been considered a mismatch in terms of Agile e.g. within [Beznosov and Kruchten, 2004] it is highlighted that the extensive documentation required runs counter to Agile practices. Thus organisations embracing Agile must carefully consider how to integrate secure development practices within their environment. Importantly, whilst Agile methods have been criticised for overlooking security assurance, [Rindell et al., 2015] believe that they are capable of being adapted to such environments, however, we found no evidence to show that software SMEs had managed to fully embrace Agile and integrate secure development practices within the context of this adoption. Therefore, given the additional complexity of achieving a balance between Agile and security (as questioned by [Beznosov and Kruchten, 2004]: how can they adopt each other), and the mass adoption of Agile, it is not surprising to find that the software V&V activities have reached a more mature level of practice within such organisations (or perhaps are less impacted by the adoption of Agile, see Section 8.4).

Therefore, although the majority of organisations report Agile adoption, we find this might be a case of 'lip service'. For example, although Agile should improve interactor communication (e.g. [Begel and Nagappan, 2007] highlight, within Microsoft, that daily scrums were instrumental in bringing developers and test engineers together), we do not find that the level of communication which exists (see Section 5.6.6) has resulted in improved relationships between the primary interactors or the erosion of their traditional roles (in some instances, such benefits cannot be directly attributed to the adoption of Agile since, within some of the organisations, the communication between the two types of interactor was already well established, see Section 6.3.7.2). Similarly, a failure to consider the integration between Agile and security practices can result in their superficial treatment i.e. a focus on hardening sprints or only in the early phases of a project [Rindell et al., 2015], with some organisations, when adopting Agile, also overlooking security early on during software development [Cao

and Ramesh, 2008]. This emphasises the need to consider security V&V throughout the software development lifecycle, regardless of the employed software development methodology (which is a frequency of practice found in terms of software V&V, but not security V&V, see Section 6.4.1.1).

Importantly, before an organisation embraces Agile, training should be provided by the adopting organisation [Talby et al., 2006, Rigby et al., 2016]. This must include both developers and test engineers and, whilst considered a significant challenge, this should include ensuring that testing is everyone's responsibility [Talby et al., 2006] (as we observed, training is often limited within SMEs, see Section 5.6.7.3). For example, [Cao and Ramesh, 2008] observed difficultly in developers embracing Test-Driven Development (i.e. writing tests before coding), thus the developers reportedly failed to consistently follow this practice. Further, the impact of placing test engineers within scrum teams, without any training, is captured by [Crispin and Gregory, 2009]: all the test engineers subsequently left their organisation - within the space of three months - due to not understanding their new roles. As Crispin and Gregory indicate, such problems can be avoided with the appropriate training. Similarly, test engineers must change their mindset to successfully transition into an Agile team [Talby et al., 2006]. There is still an uneasy relationship between the primary interactors (see Section 6.3.6.3 which, interestingly, we found is perpetuated by other interactors within an organisation, see Section 6.3.6.4), which may continue to reflect the adversarial nature of test-based activities [Cohen et al., 2004]. However, that developers and test engineers perceive software development differently [Cohen et al., 2004], and with test engineers possessing lower skills [Dhaliwal et al., 2011], this makes the sought for equality on an Agile team [Maier et al., 2017] even more difficult and challenging to achieve (this was particularly evident within one organisation, where the Lead QA Engineer [O:VS-I:GH] - who was also the Scrum Master - conveyed the inequality between developers and test engineers within their organisation echoing, by implication, that Agile, by itself, is not a panacea).

Notably, the adoption of Agile, and its associated training, must also extend further than just the engineering teams [Rigby et al., 2016]. Changing software development processes impacts organisational structure and culture, as well as management practices [Nerur et al., 2005]. Rigby et al. also emphasise that an organisation's executives require training since, without it, they will continue to manage in ways opposed to Agile, therefore, undermining its effectiveness and that of the Agile teams [Rigby et al., 2016] (this may have led to some organisations retaining traditional approaches, for example, [Theocharis et al., 2015] posit the unconfirmed hypothesis that organisations (especially large ones) use traditional processes as reference models to define manage-

ment processes, with the project teams utilising Agile methods). Thus, they see the behaviour of executives as the greatest impediment to the successful adoption of Agile (we recall the limited direct involvement of management with V&V, see Sections 5.6.4.2 and 6.3.5.3). Management involvement, during an organisation's transition to Agile, is particularly important in a V&V context, for example, how management promotes developer testing within an organisation [Talby et al., 2006] (Talby et al. indicate this is a major issue within organisations where test activities are poorly perceived which, as we found, is the case within many of the organisations studied). As one of the core principles of Agile is to build projects around motivated individuals [Rigby et al., 2016], this may explain why the adoption of Agile has not led to an improvement in security V&V. Since, aside from the observed lack of management support (i.e. with organisations not providing the environment and support necessary to perform the V&V activities, particularly so in terms of security V&V), we find, like [Perry, 2006], that a lack of management support can result in a lack of interactor motivation and interest (we found higher levels of motivation surrounding the software V&V activities, see Figure 5.19). Thus, implementing Agile not only requires a change in mindset at the individual level, but also at an organisational level [Talby et al., 2006]. Therefore, it is apparent that Agile is not a panacea, and that people must consider the adoption of Agile carefully, coupled with the investment necessary to ensure that it is not just 'lip service' adoption so as to achieve its full benefits. However, it is also clear that integrating security V&V, within an Agile framework, is a complex problem, as well as one which is only recently being explored.

7.3.2 A Lack of Awareness Surrounding Security V&V Practice

The adoption of Agile aside (which advocates co-location and centralised teams [Kongsli, 2006]), as we focused on SMEs, we found that many of their engineering teams were small and centrally located (with some mixing their developers and test engineers, due to their adoption of Agile, and with some retaining a separation, see Section 6.3.1). However, predominately, all primary interactors, within the context of an individual organisation, were situated within the same building, often the same floor, with very little in the way of spatial boundaries (reflecting the popularity of open plan offices [Bernstein and Turban, 2018]). Similarly, the majority of the secondary and tertiary interactors identified (excepting customers and the use of outsourced engineers and consultants) were also typically based within the same building. This, coupled with the adoption levels of Agile, and how central face-to-face communication is to its practice [Rigby et al., 2016], we would have expected an increased level of awareness of an organisation's security V&V practices (when practiced and, indeed, when not prac-

ticed) since, more generally, [Biehl et al., 2007] state that developers spend significant time gaining, and maintaining, a level of awareness of the activities being performed by other developers and, aside from improved communication, the main perceived benefit of adopting Agile is an awareness of other people's work [Murphy et al., 2013]. Thus, collectively, we had assumed that the proximity of engineers - and the adoption of Agile (through providing transparency of an interactor's tasks, as well as improving the frequency of interactor communication) - would have resulted in an increased level of awareness of the V&V activities within an organisation. However, we found that people were less aware of the situation regarding the security V&V activities - including as to whether they were practiced, and whether there were tools and processes supporting the activities. It was also evident, throughout the interviews conducted, that these activities were more challenging for interviewees to discuss. This suggests, at best, an inconsistency in security V&V practice, which also highlights that such practices have not reached the maturity reflected in the software V&V activities practiced by software SMEs.

Thus, whilst the Agile practitioners surveyed and then subsequently interviewed by [Murphy et al., 2013] felt that the strongest perceived benefit from Agile adoption was improved communication, followed by an awareness of other people's work, we, like [Biehl et al., 2007] (who also performed a survey followed by interviews), found that developers lack awareness of other group members' actions (they explicitly attribute this to inadequate tool support). Notably, as our respondents and interviewees included developers and test engineers, as well as those overseeing both, we cannot attribute the lack of awareness to any one type of interactor i.e. test engineers becoming sidelined (as found in [Whitworth and Biddle, 2007, Whitworth, 2008]). This, therefore, suggests a more systemic problem which, in turn, is supported by the finding that interactors are more likely to be excluded from the security V&V activities (in some instances, deliberately). This suggests a continued, enforced separation between Agile and security, as well as limited interactor interaction regarding the security V&V activities themselves. This is unfortunate, since when developers become responsible for testing, their test-awareness levels also increase (in turn, developers are then more likely to design software with testability in mind) [Talby et al., 2006]. This improved awareness can also lead to an improvement in the relationship between developers and test engineers, notably, helping reduce invalid defect reports (we recall, and as found, defects are an area of contention between the two primary interactors [Cohen et al., 2004, Vogel, 2011]). As Talby et al. further observe, such task co-ordination is often difficult when developers and test engineers are based within separate teams.

As we continue to find evidence that Agile is not a panacea - in terms of security

V&V - we also find that a lack of awareness is more pervasive than just the V&V activities themselves. For example, [Hoda et al., 2008] indicate that a lack of awareness of Agile leads to skepticism, resulting in difficulties surrounding its adoption. We found limited evidence that Agile had been embraced to the level required (i.e. there is a degree of 'lip service' adoption being exhibited within the organisations). Significantly, [Pikkarainen et al., 2012] found that a lack of awareness, surrounding Agile and its associated methods, resulted in a commitment problem, not only amongst the developers within an organisation, but also its management. Therefore, and with such a basis, the recognised difficulties of integrating security practices within an Agile software development context (see Section 7.3.1) are further compounded. However, [Kongsli, 2006] states that it is possible to improve the level of awareness, within an Agile team, regarding such practices. Specifically, they indicate that through improving the level of security awareness, amongst the team, there formed a stronger sense of ownership, whereupon members of the team became accustomed to questioning how they could misuse the product. Certainly, raising the levels of security awareness, throughout a team, helps bring security issues into the foreground [Rindell et al., 2017]. As is apparent, and regardless of the types of interactor involved, there is evidently a lack of awareness surrounding the security V&V activities.

However, looking past an organisation's adoption and use of Agile (since some organisations continue to practice other methodologies both alongside, and instead of, Agile), we still find that there is less awareness concerning security V&V than software V&V. This helps support the view that security has become 'someone else's problem' which, as [Kongsli, 2006] acknowledges, is the case within Agile environments when security is dealt with by members outside of a team i.e. the team ceases to worry about security. As we found, this attitude is more pervasive than just engineering (also encompassing an organisation's management) and is further compounded by a lack of interactor involvement with the security V&V activities (see Section 5.6.5), a lack of interactor knowledge due to the inherent complexity of these activities (see Section 6.3.6.6), as well as a reliance on the outsourcing of these activities (see Section 6.3.7.4).

Notably, our findings concerning security V&V awareness fit, and are aided by, the Information Security Competence Maturity Model (which draws inspiration from the more generic Conscious Competence Learning Matrix) [Thomson and von Solms, 2006]. Specifically, we found, overwhelmingly, that the interactors existing within the V&V socio-technical network display a limited level of awareness in terms of security V&V (both in its specific application within their organisations and, more generally, in how the activities should be applied), as well as their role in this regard. According to Thomson and von Solms, this suggests employees are primarily at the 'unconsciously

incompetent' stage. To progress to the second stage, namely, becoming 'consciously incompetent', it is necessary to increase awareness levels (Thomson and von Solms discuss this in terms of information security more generally). Hence, it becomes necessary to raise the level of security V&V awareness within UK-based software SMEs. According to Thomson and von Solms, increasing awareness should precede any training in order to help employees move to the 'consciously competent' stage (it is then, through experience, possible to move to the 'unconsciously competent' stage). Notably, given the relationship established between an organisation's information security culture, and their level of security V&V practice (see Sections 5.8.2 and 6.4.2), as well as the importance of engaging an entire organisation behind such practices (see Section 6.3.6.5 for the impact interactors outside of engineering can have on security V&V), we find that it is necessary to both raise awareness of security within an organisation, and improve its inclusivity. Awareness, in the case of security V&V, will only develop through communication. Unfortunately, we find that such communication is hindered both by its inherent complexity and its apparent interactor exclusivity. This limits the building of a shared, common knowledge, and a level of awareness resulting in team members feeling secure and comfortable, and thus increasing the sense of individual responsibility [Whitworth and Biddle, 2007, Whitworth, 2008] i.e. security does not become 'someone else's problem'. One way of improving security V&V awareness levels, within an organisation, is to nominate a security champion i.e. someone with recognised responsibility and authority in terms of security V&V. We found, where such a champion existed, that whilst others within an organisation could by no means be called experts, security had a tangible presence, with employees having someone to approach and to be guided by (such resource can help propagate security thinking throughout a team [Wäyrynen et al., 2004]). Further, a single security engineer can support 5-20 teams, if used early and appropriately [Beznosov and Kruchten, 2004]. Therefore, even given the resource constraints within SMEs, a single, focused engineer (we avoid the term 'dedicated' given that many employees may have other roles e.g. also performing the role of developer or test engineer, thereby echoing the observed multiple hats worn by employees within SMEs [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018], see Section 6.3.8.2) could act as the catalyst necessary to improve security V&V awareness within an organisation, thereby bringing people to consciously recognise their current levels of security V&V practice with the view of improving upon them.

7.3.3 Reliance on External Security V&V Expertise

Although outsourcing offers clear benefits to organisations, including within a V&V context [Karhu et al., 2007, Smuts et al., 2010], it is apparent that a combination of

the complexity of security V&V (see Section 6.3.8.7), and the resourcing constraints of software SMEs more generally, has led to a reliance on external resources performing the associated activities. This supports the view that software V&V has reached a higher level of maturity, however, it has also contributed towards the decline in internal growth and development of security V&V capability within software SMEs.

This reliance on external expertise should not be wholly unexpected given the inherent complexity of security V&V. It requires the blending of a variety of subjects and domains of knowledge and, as has been identified, this is further compounded by the identified lack of training being afforded to graduates [Kreeger, 2009]. Whilst this impacts their ability to 'hit the ground running' when starting at an SME in terms of contributing to an organisation's security V&V practice, the limited training within SMEs, especially concerning security V&V (see Sections 5.6.7.3 and 6.3.8.6), continues to perpetuate the trend. Further, and although [Grant, 2013] considers states as opposed to organisations, and military capability rather than security V&V capability, it is noted that whilst states should prefer to become independent, in terms of security, very few will actually achieve this. Grant continues that most states produce what security they can (this is considered their 'internal security') and then rely on others (their 'external security'). Notably, the states considered as having a greater advantage in terms of attaining security are those with larger populations, greater resources and improved technology - effectively, the situation one would expect to exist in larger organisations and not within SMEs (which supports the resource flows we observed within SMEs, see Sections 5.6.7 and 6.3.8).

Therefore, it is not unsurprising that organisations, when realising the need to ensure security within their products, turn towards an external agency to perform the associated activities. Unfortunately, this has several direct consequences, not least restricting the growth of an organisation's internal security V&V capability, but also, given the costs of engaging such external experts, resulting in the activities being less likely to be performed throughout the course of software development i.e. often leading to a focus towards the end of a project. It is commonly regarded that security cannot be tested into a product [Howard and Lipner, 2006, Takanen et al., 2008], with security also being something that should not be considered solely towards the end of a project. Specifically, security V&V should occur early (especially at the point when designs can be influenced), as well as throughout the software development lifecycle - regardless of the adopted software development methodology - in order to ensure any regressions are caught i.e. security should be considered an emergent property [Arkin et al., 2005]. It is worth highlighting that the situation is different in terms of software V&V more generally where, in contrast, university curricula focuses significantly

more on programming, design and, to some extent, testing (although still considerably more than security-focused testing [Kreeger, 2009]). Therefore, in this context, there is a stronger basis on which organisations can develop their employees - especially through on-the-job and self-directed learning (which reflects the observed training patterns within software SMEs [Lethbridge, 2000, Sung and Paynter, 2006], see Section 5.6.7.3).

However, whilst the desire to outsource security V&V appears to be driven by a lack of knowledge and expertise, the increased reliance on outsourcing for security V&V also appears to reflect the "insource the core and outsource the rest" view adopted by many organisational leaders [Bayrak, 2013, p. 15]. As found, security has been viewed as just another software quality attribute and has not been given the prominence it deserves - especially by an organisation's management. Thus, given the complexity of security, and the position in which it is held, outsourcing security V&V probably appears a sound strategy to many within an organisation's management team i.e freeing up an organisation's resources (e.g. their engineers) and allowing them to focus on their core competencies [Bayrak, 2013]. However, whilst developers are the most appropriate type of interactor to assume the responsibilities of a security specialist [Rindell et al., 2015], Beznosov and Kruchten, who discuss security expertise and testing philosophy, indicate such a combination cannot be expected from an average developer [Beznosov and Kruchten, 2004]. Rindell et al. also indicate that such a pairing is a violation of the 'separation of duties' rule i.e. developers are seldom best placed to break their own code. Although developers may be optimally positioned to perform security-focused testing we find, generally, that the organisation environment required to cultivate this does not exist e.g. with dedicated training programmes and the management assistance necessary to support and motivate interactors. In turn, this is likely to further reduce motivation levels when performing any form of test-based activity - even given the time to learn, or to perform, security V&V (there are also clearly hurdles to overcome in terms of motivating developers to perform test-based activities, see Section 6.3.6.2).

Therefore, although there is a reliance on outsourcing security V&V, and a trend to increase such reliance (see Section 5.6.2), we would, of course, argue that if an organisation is not yet in a position to move such activities in-house, they should continue to outsource - i.e. some security V&V is better than none - however, organisations should cease to allow the activities to be managed by outsourced agencies, bringing control back within the organisations. Specifically, to help reduce the observed 'throw over the wall' mentality between an outsourced agency and an organisation's engineering team, there are several actions that an organisation can undertake to improve the in-house situation. For example, the level of communication between the outsourced agency and

the engineering team needs to be widened e.g. the report from the outsourced agency should not only be circulated to a subset of developers, but should also include the project's test engineers. Whilst it is understandable that some restriction should be placed on security sensitive material, test engineers would greatly benefit from reading such information (similar to the need of their being involved in code review activities, see Section 5.6.6.1). Further, but likely to generate additional costs to an organisation, in some cases it is possible to request that the outsourced agency provides some training/discussion as to how they approached vulnerability discovery. Once again, this should involve both developers and test engineers in order to maximise learning potential within an organisation (and to bring different perspectives given the interactors' different skill sets and mindsets [Pettichord, 2000, Sawyer, 2001, Dhaliwal et al., 2011, Zhang et al., 2014, Zhang et al., 2018]). In addition, and in some instances, the outsourced agency may develop custom scripts/applications to assist in vulnerability discovery (e.g. a customised fuzzer). When formulating the contract with the outsourced agency an organisation should consider whether requesting access to the source of these tools is appropriate. In summary, the communication surrounding the security V&V activities - regardless of mechanism and including both that which occurs internal and external to an organisation - needs to improve.

Ultimately, however, it is important to establish, and grow, an organisation's internal security expertise (especially when considering, for example, that outsourced penetrating testing is both a relatively new and unregulated proposition, with an associated low barrier [Yeo, 2013] - this may result in some external engagements not providing the results expected, in turn, providing an organisation with a false degree of confidence in the security of their products; further, a lack of domain knowledge and commitment from external developers, cultural clashes, and poor communication, can compound the problem [Moe et al., 2014]). As noted by McGraw, developing secure software does not necessarily require "fundamental, earth shattering, or cost prohibitive" changes to an organisation's approach to software development [McGraw, 2006, p. 83]. For example, a single security engineering resource has been deemed sufficient in supporting up to 20 teams if embedded early and used appropriately [Beznosov and Kruchten, 2004]. Based on the size and structure of the engineering teams within software SMEs (see Section 6.3.1), a single, focused resource would probably be sufficient within this organisational context to provide the capability and support necessary ([Wurster and van Oorschot, 2008] consider this a more sensible approach, rather than attempting to train every developer into becoming a security expert). This is borne out by the observed reliance placed on the sole Security Test Engineer, within a medium-sized organisation [O:RV], in terms of meeting their security V&V needs (this is within an organisation

where product security is paramount for their continued operational sustainability; it is also an organisation which does not employ any form of outsourcing). Further, having established a relationship between an organisation's information security culture, and the maturity of their security V&V practice, we recommend, as found with Agile adoption (see Section 7.3.2), that organisations continue to grow their level of security awareness i.e. it should not just be an engineering problem, falling on to one or two specialised individuals to the exclusion of all others, but be propagated throughout an organisation, thereby becoming more inclusive - in turn, helping security issues take centre stage [Rindell et al., 2017]. A nominated, and recognised, internal security V&V champion can help develop this level of awareness.

7.3.4 Anticipating the External Influences on Security V&V

Whilst we have predominately focused on the factors existing within an organisation, in terms of driving, or hindering, their ability to improve their security V&V (e.g. by nominating a security V&V champion, improving their information security culture and awareness, the impact of time and the use of outsourcing), we should not discount the influencing factors existing external to an organisation. Specifically, although the improvement, and maturing, of an organisation's security V&V capability must clearly be driven internally, such a drive for change will also originate external to the organisation. For example, with customers requesting information on vulnerabilities and the V&V being performed within an organisation, to the need to meet a variety of standards and regulations. SMEs should be cognisant of these demands since, in terms of security V&V, we do not see evidence supporting the view that SMEs are either close to their products or customers [Antony et al., 2016].

Notably, software security has been considered a 'market for lemons' [Anderson, 2001]. This denotes, in terms of a product's security properties, that an asymmetry in information exists between software producing organisations and their customers. As this asymmetry changes, it is believed that the 'market for lemons' should also decrease [Ozment, 2007]. However, whilst customers are becoming more aware, and sensitive to, security issues than they were previously [Abawajy, 2014], we found that such interactors were only infrequently involved with an organisation's V&V activities, therefore, limiting their ability to influence the activities (see Sections 5.6.2 and 6.3.2). There was also limited communication, concerning the V&V activities, with customers (see Sections 5.6.6.1 and 6.3.7.4). This potentially reduces their voice in terms of requesting additional security V&V or information on what has been performed (however, this does not prevent their raising vulnerabilities or actively requesting information - it also emphasises that organisations need to fully embrace Agile to improve their

communication with customers). In terms of developing software with usable security, [Caputo et al., 2016] found that organisations were more likely to respond to pain i.e. a large number of customer complaints (both internal and external to an organisation) or from influential individuals. This implies a reactive approach to security which overlooks calls to become more proactive [Schneier, 2003, McGraw, 2006]. As found by [Kwon and Johnson, 2014], in terms of the healthcare industry, a proactive approach to security investment is more cost effective than a reactive one, corresponding with a lower security failure rate - thereby implying that organisations should not wait for either a customer to request, or complain, in order to perform an appropriate amount of security V&V (the amount of security V&V required is, of course, dependent on context e.g. where the product will be deployed and the impact of its failure). Thus, whilst customers should help drive the improvement of an organisation's security V&V capability, this should not be driven directly by them, but rather through a desire of the organisation to provide a secure product to the customer. However, without a strong incentive for an organisation to improve the security posture of their products, the 'market for lemons' may continue. This can be countered, to some extent, through vulnerability disclosure i.e. by incentivising organisations to address vulnerabilities (see [Anderson and Moore, 2006, Arora et al., 2008]), or by requiring conformance to a variety of security relevant standards and regulations.

Notably, there are many standards and regulations governing information security [Chakraborty and Raghuraman, 2015] - certainly, we found a variety referenced and being adhered to (see Sections 5.7.3, 6.3.5.4 and 6.4.2). However, the limited number of organisations doing so may reflect the view of [Janczewski, 2000], who questions, for example, whether Common Criteria is feasible for smaller organisations. Similarly, [Anderson, 2001] considers Common Criteria in a somewhat negative light. However, Anderson also observes that in some sectors such certification offers competitive advantages. Whilst we would not challenge this view, we feel that these standards provide the necessary impetus for an organisation to improve their security V&V practice. Thus we view such standards as a means of improving the security posture of their products by forcing organisations to meet certain standards so that they can enter specific markets. However, as is apparent, even meeting some of these standards does not guarantee a secure product (e.g. [Deeg and Schreiber, 2009]) - although we continue to maintain the view that they offer some level of guidance and focus to an organisation. Notably, such focus is more likely to galvanise an organisation's senior management into supporting their organisation's security V&V practice (i.e. echoing the observed desire to generate additional sales, rather than creating a secure product per se).

As observed, management support, and their level of understanding regarding se-

curity V&V, was low within the organisations studied (this was the case both in terms of support (e.g. a lack of training, see Sections 5.6.7.3 and 6.3.8.6) and motivating interactors into performing security V&V, see Sections 5.6.4.2 and 6.3.5.3). Whilst it is unnecessary for an organisation's management to become recognised as security experts, having some comprehension is required to understand why they should support the activities (e.g. through training, the hiring in of expertise, the purchasing of tools and making the time available to perform the activities). Notably, being forced to comply to a piece of regulation, in order to sell to a specific country or market, helps provide that level of focus and understanding. Although organisations should be motivated to produce secure products without being driven by such profit-driven reasoning, we feel ensuring the security of a product - regardless of motivation - is preferable to not ensuring security. However, this, once again, is where improving an organisation's information security culture makes such conversations between engineering and an organisation's management easier, since there is then a common language (we recall that the complexity of security is well-known [Kreeger, 2009, Austin and Williams, 2011] - which can lead to difficulties when discussing such a complex subject e.g. it can overwhelm both security experts and managers [Wang and Guo, 2009], thus probably explaining why books titled "Selling Information Security to the Board: A Primer" have started to exist [Calder, 2016]). Ultimately, whilst compliance with standards and regulations may also be considered reactive in nature (e.g. they are typically formed in response to a specific need, which may subsequently have passed) [Duncan and Whittington, 2014], it is also apparent that they will not be adopted by SMEs unless they are known to them. Therefore, organisations should cultivate an awareness of the relevant security standards and regulations, and the demands of their customers - and future customers - before initiating a project (or, at least as early as possible) to avoid security becoming an expensive afterthought.

7.4 Conclusions and Reflections

Within this section we revisit the research objectives in light of the synthesis and discussion performed. We then reflect on how we initially conceptualised V&V as a socio-technical network and how our understanding changed following two phases of data collection and analysis.

7.4.1 Security V&V Practice within SMEs

After two phases of data collection and analysis, we believe that we have successfully acquired an understanding as to how security V&V is practiced, supported, and per-

ceived, within a diverse set of UK-based software SMEs. This was important, since there was a general dearth of existing knowledge concerning the socio-technical realities surrounding how these activities are practiced within software SMEs. For example, where aspects of security V&V were touched upon in the existing literature, they either focus on the technical, to the obvious detriment of the social ([Rooksby et al., 2009] observe that the software testing literature is too focused on technical issues, thereby overlooking the fact that V&V encompasses a set of socio-technical activities, see Chapter 3); or they would present results in terms of a single activity (for example, just security testing e.g. [Cruzes et al., 2017], whereas our definition of what constitutes security V&V was based on a series of established software security maturity models (i.e. [McGraw et al., 2016, OWASP, 2017]) and encompasses more than a single type of activity), or they failed to distinguish between small and large organisations (e.g. [Geras et al., 2004, Garousi and Varma, 2010, Larusdottir et al., 2010, Cruzes et al., 2017], however, such organisations have long been known to differ [Street and Meister, 2004]). As such, many existing studies fail to address the specific organisational and technological contexts defined by this thesis (see Sections 1.2 and 1.3 respectively), as well as often failing to account for both the social and the technical realities which surround such practice. However, whilst we do not posit that we have reached the end of our journey in understanding security V&V practice within software SMEs (see Section 8.4), we believe that the level of knowledge acquired is comparable to that of many existing empirical software engineering studies - including those focusing upon V&V more generally and especially those touching upon aspects of security V&V.

In summary, we have identified the interactors involved across the security V&V activities, including those which are human and non-human, as well as technical and non-technical (i.e. Steps 1 and 2 within STIN). However, unlike many existing studies, we elected not to focus on just developers and test engineers since, as found, many other types of interactor exist within the V&V socio-technical network and either directly participate in the activities, or have influence over them or the primary interactors. We also obtained an understanding of what motivates the identified interactors into performing the activities (Step 3), as well as how the interactors communicate with one another (Step 5). Further, by identifying resource flows (Step 6) and the undesired interactions surrounding the activities (as well as excluded interactors i.e. Step 4), we were able to determine what influences the activities being performed within an organisation, as well as the factors which exist external to an organisation but ultimately influence the activities within. The level of understanding obtained was greatly supported through the adoption of STIN, which not only helped focus our attention on aspects typically overlooked (e.g. undesired interactions and excluded actors [Meyer,

2007]), but also enabled us to identify patterns of routine use. Both were essential in obtaining a more complete account of how security V&V is practiced within software SMEs (see Chapter 3).

Based on the findings obtained, we are now in a stronger position to identify areas warranting additional research (see Section 8.4), as well as areas where security V&V practice can be improved (see Section 8.3.2). For example, software SMEs should focus on improving the level of security V&V awareness within their organisations, thereby preventing security V&V being continually seen as 'someone else's problem' (see Section 7.3.2). As we found, it is important that awareness levels increase throughout an organisation and not just amongst the primary interactors (i.e. those performing the activities) since other interactors, in particular, management, can strongly influence the activities (not only in terms of financial support, but also, through their support more generally, by helping convey that the activities have worth). Similarly, organisations should seek to improve their level of in-house security V&V expertise (so as to become more effective in their application, but also in sharing a common language with others, both those internal and external to an organisation). Ideally, organisations should nominate a security V&V champion. Given the appropriate authority (i.e. visible management support), such individuals can continue to increase the levels of security V&V awareness within an organisation and help propagate security knowledge throughout. As we and others observe (e.g. [Beznosov and Kruchten, 2004]), a single resource of this nature can make a significant difference within an organisation.

Therefore, as customers continue to show greater levels of interest in the security properties of the products they are purchasing and deploying [Zabicki and Ellis, 2017], software producing organisations will need to become less reactive to security and much more proactive in nature (e.g. security should no longer be seen as 'someone else's problem' or just something which can be added towards the end of a project in order to enter new markets so as to make some additional sales). Notably, we, like others (e.g. [McGraw, 2006]), find that improving a product's security posture does not need to be cost prohibitive to an organisation, therefore, we believe it is realistic for software SMEs to improve their security V&V practice. Given the insights obtained, there is evidence to suggest that such improvements are necessary, which is further supported when observing that the number of vulnerabilities being reported continues to increase, with an evident trend that the same security-related mistakes continue to be repeated [Piessens, 2002, McGraw, 2006, McGraw, 2008, Anderson, 2008, Jourdan, 2009, NIST, 2020].

In conclusion, although understanding how UK-based software SMEs perform security V&V was previously considered an empirical unknown (see Section 1.3.6 and

[Kreeger and Harindranath, 2012]), we believe, after two phases of data collection and analysis, that this is now less of an unknown (see Chapters 5 and 6 and [Kreeger and Harindranath, 2017]).

7.4.2 The Influence of Information Security Culture on Security V&V

By establishing the security V&V maturity level within a software SME (see Section 7.4.1), and then complementing this with an understanding of the strength of their organisation's information security culture - which we recall should impact an employee's daily activities [Schlienger and Teufel, 2003, Sánchez et al., 2010] - we are then able to gauge whether such an organisational culture can reach, and pervade, an organisation's software development lifecycle. Effectively, we can determine whether an organisation's information security culture influences their security V&V practices, as well as whether this influence is positive or negative.

Notably, we observe that an organisation's information security culture has typically been considered in isolation from the products they produce. Further, many information security culture-based studies overlook the unique characteristics of SMEs, especially within a national context [Dojkovski et al., 2007, Dojkovski et al., 2010]. However, although the specific organisational and technological contexts of this thesis have not been explicitly addressed within the existing information security culture literature, there exist a variety of instruments and guides upon which to base our own approach. Specifically, we developed our survey instrument, and interview guide (see Appendices H and K respectively), around the OECD's principles and guidelines for developing a security culture within an organisation (i.e. [OECD, 2002, BIAC and ICC, 2003] - which are relevant to SMEs [BIAC and ICC, 2004]). We also made reference to several other, relevant studies (e.g. [Schlienger and Teufel, 2003, Burns et al., 2006, Ngo et al., 2009]) to ensure that the main elements of assessing information security culture were covered (e.g. policies, awareness, training and education [Dojkovski et al., 2007], see Appendix D).

Having identified the interactors involved with an organisation's information security more generally (Steps 1 and 2 within STIN), we were then able to identify who was responsible for information security within an organisation, including the artefacts supporting them (e.g. an organisation's information security policy which, as we found, could also be used to motivate the identified interactors - thereby also covering Step 3). Interestingly, and on comparison with [Dimopoulos et al., 2004] (who also focused on information security culture within SMEs), we found a greater level of known, and recognised, responsibility regarding information security governance within the SMEs we studied (see Section 5.7.1). We also found that those responsible typically held

other positions, therefore, supporting the view that employees within SMEs often wear multiple hats [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018]. Additionally, and even recognising the importance of possessing an information security policy [von Solms, 2001b], other SME-focused studies have suggested that such organisations do not typically have formally documented information security policies (e.g. [Dimopoulos et al., 2004, Burns et al., 2006]). However, we find a somewhat healthier picture (see Section 5.7.2), whereupon, we also find that organisations possessing an information security policy are more likely to have processes governing their security V&V activities (we recall that many maturity models consider a governing process as an indication of a practice with a higher associated maturity level e.g. [CMMI Product Team, 2010, Kollanus, 2011], see Section 5.8.2.2). These examples demonstrate the importance of performing studies within specific contexts e.g. within software SMEs, since many studies do not consider a specific industry sector (e.g. [Burns et al., 2006], who consider a variety of Welsh SMEs) or national context (e.g. [Dojkovski et al., 2007, Dojkovski et al., 2010]). Thus we were also able to position our information security culture findings within a software SME-focused, UK-based practitioner context.

In conclusion, we find that the presence of an organisation's information security culture does impact an organisation's security V&V practice, specifically, in a positive way (see Sections 5.8.2 and 6.4.2). This is particularly evident in terms of the resource flows surrounding the associated activities (Step 6 within STIN), whereupon, we find that there is a significantly greater level of support being afforded to the security V&V activities when an information security culture is present within an organisation (e.g. in terms of the tools used, the automation levels attained and the level of training provision provided, see Sections 5.7.6 and 5.8.2.4). We recall that the purpose of an organisation's information security culture is to ensure that security becomes ingrained in an employees' daily activities [Schlienger and Teufel, 2003, Sánchez et al., 2010, Okere et al., 2012]. This implies that a degree of awareness of security V&V is required by employees, which is something that we identified is generally lacking within software SMEs (see Section 7.3.2). However, where a strong information security culture is found to exist, we find healthier levels of awareness and support given over to the security V&V activities (certainly, relative to the organisations where there is a weak, or a non-existent information security culture). Notably, the security V&V activities are practiced with more regularity within the software SMEs reporting an information security culture (see Section 5.8.2.1). Further, by encompassing the entire organisation, we find that security is less likely to become 'someone else's problem' (which was something we, and others, observed e.g. [Gokhale and Banks, 2004], see Section 7.3.2) since it becomes a shared responsibility. Additionally, and acknowledging the need for

organisations to enforce their information security policies [von Solms and von Solms, 2004, von Solms and von Solms, 2009], we find that those which follow this practice are more likely to report a stronger information security culture - which, as we also found, positively influences security V&V practice. Therefore, we find that improving an organisation's information security culture helps support the continued development of an organisation's security V&V capability.

7.4.3 The Relationship Between Software and Security V&V

By examining how UK-based software SMEs practice, support, and perceive, software and security V&V, we were in a stronger position to better understand the relationship between the two sets of activities. In particular, we were able to see where a shared heritage existed (e.g. a common set of tools applicable to both software and security V&V), how organisations emphasised particular activities (to the potential detriment of others), as well as whether maturity levels, across the two sets of activities, were synchronised or not. Notably, and given the inherent resourcing constraints of SMEs [Sitnikova et al., 2007, Cruz-Cunha, 2010], and the importance of security V&V (see Section 1.3.6), it was important to determine whether organisations could achieve either a comparable, or a more mature, security V&V practice than their existing software V&V practice.

To achieve this, it was necessary to ensure that a consistent approach was taken to establishing an organisation's level of maturity across both sets of activities. As such, we recall that a variety of existing maturity models were used in which to construct an approach suitable for assessing an organisation's V&V maturity levels. Specifically, we utilised a combination of software V&V and software security maturity models (i.e. [Koomen and Pol, 1999, de Vries et al., 2009, Kollanus, 2011, McGraw et al., 2016, OWASP, 2017, TMMi Foundation, 2018], see Appendix M) which, collectively, encompass our definitions of software and security V&V (see Sections 1.3.1 and 1.3.5 respectively). Significantly, several of these maturity models have also been employed in other empirically-based software engineering studies e.g. [Koomen, 2002, Grindal et al., 2006a, Grindal et al., 2006b, Park et al., 2008, Araújo et al., 2013, Camargo et al., 2013, Camargo et al., 2015]. However, whilst several of these explicitly focus on software V&V maturity within an organisation, they often overlook an organisation's security V&V practices (which, once again, helps emphasise that security V&V practice, within software SMEs, is an empirical unknown, see Section 1.3.6 and [Kreeger and Harindranath, 2012]). Or, they focus on software security - encompassing elements of security V&V - but then avoid engaging much with software V&V more generally. Also, and as before, we typically find that a mixture of organisations are included in

these studies (thereby failing to account for the specific organisational and technological contexts of this thesis, see Sections 1.2 and 1.3). It was thus necessary to create our own survey instrument and interview guide (see Appendices H and K), but which were based on established maturity models, and existing studies, and then tailored to address our specific study context (see Appendix D).

Notably, and across the key themes of STIN, we found that the software V&V activities had typically reached a higher level of maturity within UK-based software SMEs in contrast with the corresponding security V&V activities. For example, this was exhibited in terms of interactor diversity (Steps 1 and 2 within STIN, which also emphasised the observed exclusivity associated with the security V&V activities, thereby also encompassing Step 4), the communication ecology surrounding the activities (Step 5) and the surrounding resource flows (Step 6, which exhibited itself not just in terms of increased levels of monetary support, but also in terms of expertise, associated tool support and attained automation levels). Collectively, these helped show that the software V&V activities had reached a higher level of maturity. Interestingly, and despite their shared heritage, we find that the security V&V activities sit at a lower level of maturity. For example, some tools are utilised across both sets of activities e.g. *Coverity* is used for software and security-focused code reviews, *JUnit* for software and security-focused test-based activities and *Jira* is used for capturing defects and vulnerabilities. Further, and aside from sharing elements of surrounding infrastructure, in terms of the activities themselves, we recall their similarities, for example, in terms of software and security-focused testing, both require engineers to identify resources, prepare test specifications, execute the tests, and then record and analyse the results.

However, and emphasising that the undesired interactions and resource flows within the V&V socio-technical network generally have a greater impact on security V&V (Steps 4 and 6 within STIN), we find that some of the tools which could be shared between the activities are not applied as well in terms of the security V&V activities (e.g. as observed, *Jira* is not employed as effectively in terms of tracking vulnerabilities, see Section 6.3.7.3). It is also apparent that several tools which could be classed as shareable between the two sets of activities are currently only employed in terms of the software V&V activities (this was the case with some of the xUnit test frameworks being utilised e.g. *NUnit*). Notably, the majority of the tools identified as being used for the security V&V activities were either also found to be used for the software V&V activities, or they were not transferable to such a context. This helps highlight the specialised nature of security V&V and also supports the observed lack of surrounding awareness and expertise (e.g. in some instances appropriate tools exist, but they are not utilised, see Section 6.3.8.4). It can also be partially attributed to an organisation

focusing on a product's primary functionality (see Sections 6.4.1.1 and 6.4.1.2), which implies a focus on software V&V over security V&V.

In conclusion, we did not find, across any of the organisations studied, one which possessed a security V&V practice more mature than their software V&V practice. Notably, within the majority of cases, there was a noticeable gap between the maturity levels of these related, but distinct practices. This was the case regardless of the type of organisation and their product focus (we recall the diversity of organisations reached, see Section 5.4.1, as well as that many developed software intended for deployment within environments of high security criticality e.g. within a healthcare context). Therefore, we found no evidence suggesting that organisations could possess a mature security V&V practice without a similarly mature software V&V practice. Further, and even given their shared heritage, with the current dearth of security V&V awareness and expertise within software SMEs, it is unlikely that a mature security V&V practice could develop within an organisation without an existing mature software V&V practice.

7.4.4 A Reflection on the Conceptual Model

Having performed two phases of data collection and analysis (Chapters 5 and 6), and having synthesised the results and engaged with several underlying discussion themes (Sections 7.2 and 7.3 respectively), we take the opportunity to revisit the developed conceptual model (see Figure 3.2). The revised model is captured within Figure 7.1.

Notably, we find that we initially presented a view which suggested that only developers and test engineers were directly involved with the V&V activities. Whilst this reflected the general day-to-day reality observed, it did not fully represent the situation encountered. Namely, we found much greater interactor diversity within the V&V socio-technical network, with each interactor displaying differing levels of engagement, and influence, across the V&V activities. We partially attribute this to our not fully accounting for the prevalence of interactors, within SMEs, wearing multiple hats and performing multiple roles (e.g. [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018], see Section 6.3.8.2). However, it is also reflective of the narrow focus found within many existing V&V studies (see Chapter 2). This demonstrates how easy it is to follow a well-trodden path and to just focus on the prominent and powerful interactors encountered (others have indicated similar when adopting socio-technical approaches such as ANT and SCOT e.g. [Winner, 1993, McLean and Hassard, 2004, Meyer, 2007, Mitev, 2009, Mwenya and Brown, 2017]). However, STIN, which focuses on the routine and the ordinary, helped surface the lesser involved, and excluded, interactors, as well as the surrounding undesired interactions [Kling et al., 2003, Meyer, 2007, Suri, 2013, Mor-

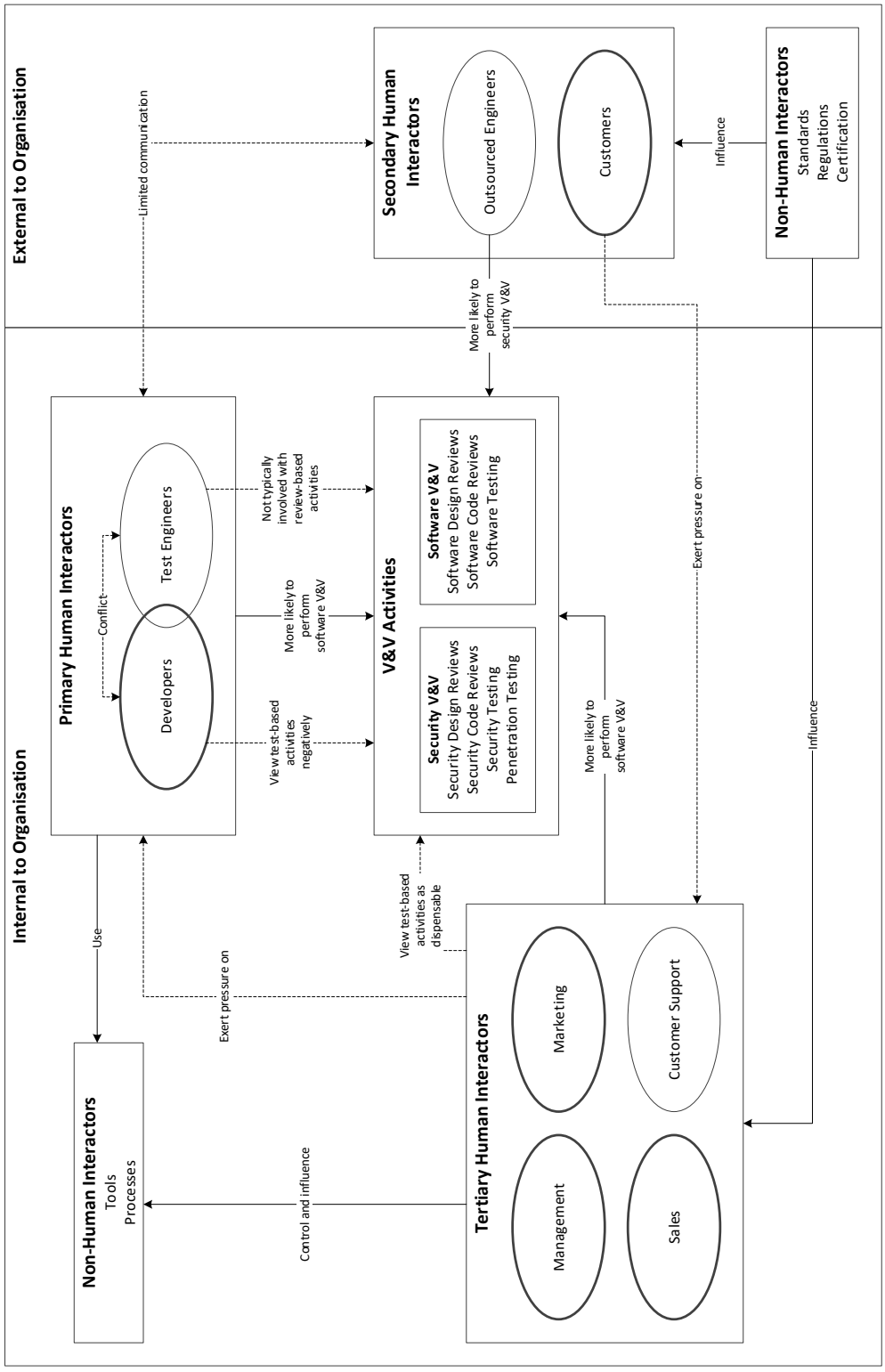


Figure 7.1: Reconceptualising V&V as a Socio-Technical Interaction Network

rison, 2014, Taylor-Smith, 2016]. Thus, we became aware of our initial focus on the primary interactors and, like [Mitev, 2009], acknowledge that some actors are 'bigger' than others (i.e. they are not all the same). Therefore, whilst the primary interactors deserved significant treatment, it would have been unwise to have overlooked the influence of the other interactors within the V&V socio-technical network i.e. the identified secondary and tertiary interactors. This not only helped convey greater interactor diversity, it ultimately helped present a more complete picture. For example, by showing that some of the interactors (e.g. the tertiary interactors), who were less directly engaged with the V&V activities, actually had more influence over the activities than many of the more directly engaged interactors, as well as the primary interactors themselves (e.g. see Sections 6.3.6.5 and 6.3.8.8).

Further, we found it necessary to differentiate between the different V&V activities within the revised model (e.g. software and security V&V, test and review-based activities). This was necessary to capture the differing levels of engagement, support, and awareness, which surround the individual activities. Additionally, and given the evident popularity of Agile (e.g. [Clarke and O'Connor, 2013, Ryan and O'Connor, 2013, Rindell et al., 2015, Chóliz et al., 2015, Rindell et al., 2017], see Figure 5.4), we had expected the distinction between the primary interactors to have become more blurred [Meszaros and Aston, 2007, Crispin and Gregory, 2009], thereby lessening the potential for interactor conflict. However, we found, within all organisations studied, that the traditional role divide between these interactors continued to persist (even in the face of Agile adoption, see Section 7.3.1) e.g. developers continue to focus on review-based activities, whilst test engineers continue to focus on test-based activities. Notably, the distinction between these interactors is also reflected in how they are viewed by other interactors within the V&V socio-technical network (see Section 6.3.6.4) which, once again, emphasises the importance of considering both technical and non-technical interactors, as well as those directly and indirectly engaged with the V&V activities.

The initial model also failed to account for an organisation's external environment. As we found (see Sections 7.3.3 and 7.3.4), this overlooked several important aspects. For example, the influence of customers (see Sections 5.6.4.5, 6.3.5.4 and 6.3.5.5) and the reliance organisations place on outsourced engineers (see Sections 5.6.2 and 6.3.2). Specifically, whilst we found that customers help motivate organisations into performing V&V, outsourced engineers, due to a lack of awareness and expertise existing within the organisations themselves, make it possible to perform these activities. This was particularly evident in terms of security V&V (see Section 7.3.3). Therefore, by initially adopting an organisation-centric focus (Meyer also reflects on the inherent organisa-

tional bias of STIN [Meyer, 2006, Meyer, 2007]), we overlooked the influences and resource flows which exist external to an organisation. However, as we analysed the data - across both phases of data collection - it became readily apparent that organisations typically focus on their product's primary functionality unless customers - either directly or indirectly - required a focus on security (e.g. with the organisations having to meet external standards, regulations, or certification, in order to be able to enter new markets). Additionally, as software security has been considered a 'market for lemons' [Anderson, 2001], it was, understandably, initially tempting to view customer influence on an organisation's security V&V practices as being somewhat limited at best (and thus, like other less prominent interactors, potentially overlook them). However, we, like others (e.g. [Zabicki and Ellis, 2017]), have since found that customers are becoming noticeably more interested in security.

In summary, we found that the impact - both positive and negative - from the interactors situated external to an organisation (e.g. the outsourced engineers, customers, standards, regulations and certification), was much greater than initially expected and which was conveyed within the initial model developed. Because of the strength of these external factors, it was necessary to revise the conceptual model in order to distinguish, and to reflect, the relationships between an organisation's internal and external elements it terms of its V&V practices. It was by including an organisation's external environment that we were in a position to develop a more holistic understanding (thereby echoing [Griffith and Dougherty, 2002]).

Chapter 8

Conclusions

Within this chapter we summarise the thesis contributions and the implications for theory and practice. We then examine and discuss potential limitations and future research directions. Specifically, this chapter provides:

Section 8.1: We reintroduce the thesis research question, associated research objectives and the approach taken.

Section 8.2: The contributions of the thesis are stated.

Section 8.3: The theoretical and practical implications of the research are presented.

Section 8.4: Limitations and future research directions are discussed.

Section 8.5: A personal reflection is given.

We begin by providing an introduction.

8.1 Introduction

Within this thesis we aimed to address the following research question: "*Within UK-based software SMEs, how is security V&V performed, and how, as a process, is it influenced by the socio-technical characteristics of such organisations?*" (see Section 1.4.1). This, in turn, was decomposed into three research objectives: firstly, "*To examine how security V&V is practiced, supported, and perceived, within UK-based software SMEs*"; secondly, "*To determine whether organisational information security culture influences security V&V practice*"; and, thirdly, "*To establish whether organisations can possess a mature security V&V practice without a correspondingly mature software V&V practice*" (see Section 1.4.2).

In order to address the above, and through acknowledging that V&V is a complex socio-technical activity, it was necessary to adopt a socio-technical approach when framing the study (see Chapter 3). We elected to use STIN (which has previously been employed when studying aspects of software engineering e.g. [Scacchi, 2005, Amrit, 2008, Jensen, 2010]). Further, and given that the combined organisational and technological contexts were an empirical unknown (see Section 1.3.6 and [Kreeger and Harindranath, 2012]), a mixed methods approach to data collection was desirable (e.g. [Mandić et al., 2009, Venkatesh et al., 2016], see Sections 4.2 and 4.6). In particular, adopting a sequential, explanatory design (an approach successfully employed in other software engineering and V&V-related studies, see Section 4.2.3), proved invaluable when undertaking research into what is undoubtedly a highly sensitive subject (this became evident as the research progressed, see Section 5.2).

In summary, having performed three phases of data collection and analysis (Chapters 5, 6 and 7), we now highlight the resultant contributions.

8.2 Contributions

Within this section we summarise the resultant contributions, which span both theory and practice.

8.2.1 Theoretical

Acknowledging the lack of socio-technical studies concerning software V&V practice in general (for example, [Rooksby et al., 2009] indicate that the software testing literature is too focused on the technical), and given the smaller number of security V&V-focused studies in existence (see Section 1.3.6 and Chapter 2), it is, therefore, unsurprising that little literature exists concerning the socio-technical realities of such practice. Specif-

ically, and recalling the composition of a typical socio-technical system (see Figure 3.1), there is a tendency for many software engineering studies to focus almost exclusively on the technical system (i.e. the tasks and the associated technologies). This view is supported by many of the existing studies concerning V&V practice. However, socio-technical theory stresses the inseparability of the social and the technical, thereby ensuring that the social system (which includes people and structure e.g. the systems of communication and authority) is also included [Griffith and Dougherty, 2002, Morris, 2009, Nelson and Quick, 2013]. This is important, since overlooking the social, and the human aspects of software engineering, will result in only a partial understanding forming (for example, just focusing on the tools used, as opposed to also seeking to understand what impacts their use within an organisation).

Therefore, whilst this study reinforces the benefits of embracing a socio-technical approach so as to derive a more holistic understanding of a situation - in this case, how security V&V is practiced, supported, and perceived, within UK-based software SMEs, and how the socio-technical characteristics of such organisations influence this practice - it is the adoption and application of STIN to this specific context which provides our main contribution to theory. Notably, whilst there is a growing body of literature showing the application of STIN across a variety of 'problem spaces' (e.g. [Letch and Carroll, 2007, Meyer, 2007, Reinert, 2009, Walker and Creanor, 2009, Urquhart and Currell, 2010, Taylor-Smith, 2016, Bingham-Hall, 2017], see Section 3.3.6), in terms of studying aspects of software engineering, the use of STIN is restricted to a few existing studies (e.g. [Scacchi, 2005, Amrit, 2008, Jensen, 2010]), none of which focus on V&V. However, having now applied STIN across three phases of data collection and analysis (Chapters 5, 6 and 7), and having addressed the identified research objectives, we highlight the value of framing an empirical V&V-focused study as a socio-technical interaction network. Specifically, we found the balance between the social and the technical systems (STIN stresses the inseparability of the social and the technical [Kling et al., 2003, Urquhart and Currell, 2010]), the inclusion of undesired interactions and the identification of excluded interactors, as well as the focus on routine use, particularly well-suited to addressing an empirical unknown in the software engineering literature. However, aside from helping reinforce the successfully application of STIN in terms of a particular 'problem space' (i.e. our specific organisational and technological contexts), we also contribute to the STIN literature more generally by discussing some of the resulting implications from its use (see Section 8.3.1).

8.2.2 Empirical

Although we discussed the growing maturity of empirical software engineering (see Section 4.1), we found, after engaging with the existing literature, a lack of understanding concerning security V&V practice generally, as well as, specifically, within UK-based software SMEs (see Section 1.3.6 and Chapter 2). In particular, we found that the majority of existing V&V studies would either focus on the technical, to the complete exclusion of the social (thus echoing, for example, [Rooksby et al., 2009] - we recall that software engineering, and security, are complex social-technical disciplines [Sawyer, 2004, Henry, 2005, Sjøberg and Grimstad, 2010, Sedano et al., 2017, Coles-Kemp and Hansen, 2017], see Section 3.2), or they avoid engaging with any aspect of security V&V (the importance of adopting a security perspective was justified within Sections 1.3.4 and 1.3.6). However, even within the studies where there is some focus on security V&V, this is usually in terms of a specific activity (for example, security testing, but not security design or code reviews e.g. [Geras et al., 2004, Causevic et al., 2009a, Causevic et al., 2009b, Garousi and Varma, 2010, Larusdottir et al., 2010, Cruzes et al., 2017]) i.e. they fail to adopt a more encompassing and complete definition of security V&V (whereas we base our definition on a series of established software security maturity models i.e. [McGraw et al., 2016, OWASP, 2017]). They also appear to avoid any coherent discussion in terms of SMEs, for example, either by failing to distinguish between small and large organisations (even though such organisations have long been known to differ [Street and Meister, 2004]) or by adopting non-standard definitions e.g. [Geras et al., 2004, Epstein, 2009, Garousi and Varma, 2010, Larusdottir et al., 2010, Cruzes et al., 2017]. Similarly, such practice within the UK, in terms of national context, also appears to have been overlooked. Thus, in summary, the extant literature fails to present a holistic account for how security V&V is practiced, supported, and perceived, within UK-based software SMEs. Having performed a three-phased approach to data collection and analysis, encompassing both quantitative and qualitative data (see Section 4.2.4), we believe that our socio-technical account of such practice contributes to the existing empirical software engineering literature. We believe this is achieved by highlighting a software engineering practice which was, hitherto, an empirical unknown (see Section 1.3.6 and [Kreeger and Harindranath, 2012]); by providing a socio-technical account of such practice; and, more generally, by offering a more complete account of such practice as found within software SMEs, based within the United Kingdom, by virtue of framing security V&V practice as a socio-technical network (i.e. by identifying the interactors involved and their complex relationships). This forms the focus of the first research objective. In summary, we found that security V&V is not performed with the regularity that we would have

expected, especially given the types of software being developed and the contexts in which they will be deployed. Aside from being performed less frequently, the activities also suffer from a lack of inclusivity, as well as a lack of understanding and awareness - thereby impacting the efficiency and confidence in which the activities are performed. This results in organisations relying heavily on external interactors to both drive and perform security V&V-related activities.

Further, and by extension, we also contribute to the existing literature on software V&V practice more generally. This is important given its evident value (see Section 1.3), as well as the repeated calls for additional V&V-focused empirical studies e.g. [Ellims et al., 2004, Kollanus and Koskinen, 2006, Itkonen et al., 2009, Larusdottir et al., 2010, Engström and Runeson, 2010, Gren and Antinyan, 2017]. However, in this instance, it is less the understanding obtained regarding practice per se, but more the socio-technical nature of the practice within a specific organisational and national context (i.e. UK-based software SMEs) which forms the contribution. However, and given the common understanding obtained of software and security V&V practice, we are also able to understand the relationship between these related - but distinct - practices. This should be seen to form a further contribution to knowledge and encompasses the third research objective. Specifically, we found no evidence to suggest that an organisation could possess a mature security V&V practice without a mature software V&V practice.

Similarly, and acknowledging that many information security culture-based studies have overlooked the unique characteristics of SMEs, especially within a national context [Dojkovski et al., 2007, Dojkovski et al., 2010], we believe that our study has made a contribution to the existing empirical information security culture literature by further exploring information security culture within SMEs and within a specific national context. However, we have also focused on a specific type of SME, namely, software SMEs. As such, and unlike many previous studies, we examine information security culture within a specific industry sector (for example, whilst [Burns et al., 2006] also focus on SMEs, based within a specific national context (i.e. Wales), they do so within a variety of industry sectors which remain undistinguished within the resulting analysis). Therefore, whilst we consider the organisational context (i.e. the type of organisation, the national context and the industry sector), as defined within Section 1.2, a contribution; more specifically, we posit the finding which shows that a relationship exists between an organisation's information security culture, and their security V&V practice, as a further contribution i.e. we find that an organisation's information security culture does pervade their software development lifecycle. This encompasses the second research objective.

We discuss the implications of these findings within Section 8.3.2.

8.2.3 Methodological

Whilst surveys and interviews have long been applied to the study of various aspects of software engineering (see Sections 4.2.4, 4.3 and 4.4), they have typically been employed in isolation from one another. Therefore, and although the concept of mixed methods has existed for a while (see Section 4.2.2), we acknowledge that their application, within a software engineering context, appears a more recent phenomena e.g. [Mandić et al., 2009, Di Penta and Tamburri, 2017]. As such, we, like others (e.g. [Storey et al., 2008, Chung et al., 2010, Tahir and Ahmad, 2010, Larusdottir et al., 2010, Deak and Stålhane, 2013, Rungi and Matulevičius, 2013, Pham et al., 2017, Zheng et al., 2017]), highlight the value in their being applied together when studying software engineering empirically. However, we highlight this value in terms of conducting a socio-technical study, focused on a specific aspect of software engineering - namely, security V&V - when employing a sequential, explanatory design. In particular, we found that the adopted research design (see Section 4.2.3), was essential for studying a highly sensitive subject. Specifically, the anonymity afforded by the initial survey enabled us to identify a number of themes, structured around the key themes within STIN, which could then be further explored within the subsequent interviews. Given the inherent costs of interviews, and the difficultly in arranging them [Ahrens and Dent, 1998, Qu and Dumay, 2011] (which is something we also encountered), we venture that the access obtained to the interviewees was only possible by virtue of having already established contact with individuals in the previous phase of data collection (thus the survey acted as a sampling frame for the interviews [Brannen, 2005]).

However, our study should not only be seen as contributing to the mixed methods literature more generally (i.e. supporting its increasing application in terms of studying aspects of software engineering empirically, as well as highlighting the value which can be obtained through employing a sequential explanatory design when researching a sensitive subject), it should also been seen to support its use when conducting a STIN-based study. Notably, whilst Meyer indicates that there are not any specific research methods tied to the adoption of STIN (although it is noted that the methods which can be used to gather data for a STIN analysis are expansive), it is observed that the research methods which should be used must be more clearly articulated [Meyer, 2006, Meyer, 2007]. Meyer considers this a practical challenge still to be overcome; and, whilst we observe that a similar approach to our study was taken within [Kim, 2008] (although our organisational and technological contexts differ), the explicit use of mixed methods was downplayed (however, they also began with a web-based

survey and then employed semi-structured interviews). Therefore, and given that such concrete guidance still does not exist, we can posit that the two well-tested approaches to data collection, namely, a survey and semi-structured interviews, when employed sequentially, worked well within our study context - bringing many of the benefits of mixed methods (e.g. [Greene et al., 1989, Bryman, 2006, Zheng et al., 2017]) to a STIN-framed study. We thus actively encourage other researchers to adopt a pluralist paradigm approach when studying aspects of software engineering empirically since, as we found, it is then possible to form a more complete picture of a given situation (thereby echoing the findings of others e.g. [Mingers, 2001, Johnson and Onwuegbuzie, 2004, Ivankova et al., 2006, Creswell and Plano Clark, 2011, Teddlie and Tashakkori, 2012, Venkatesh et al., 2013, Venkatesh et al., 2016]).

8.3 Implications

As many empirical software engineering studies "fail to do anything with their results", it is important to relate what has been learned to theory and practice [Perry et al., 2000, p. 349]. Thus we now detail the implications for theory and practice.

8.3.1 Implications for Theory

Having acknowledged the lack of socio-technical studies concerning software V&V practice in general, and given the smaller number of security V&V studies, it is understandable that little literature exists concerning the socio-technical nature of such activities (see Section 1.3.6 and Chapter 2). However, it was through embracing a socio-technical approach that we were able to obtain a more holistic view in how security V&V is practiced, supported, and perceived, within UK-based software SMEs. Thus, and referring back to Figure 3.1, we find that the concept of the socio-technical system applied well to the study of security V&V. Specifically, we find that people (the human interactors, including developers, test engineers and managers, all of whom interact with one another), utilise technology (the non-human interactors and resource flows within the network, thus including a variety of tools, techniques and knowledge), to perform a set of tasks (the V&V activities) within an organisational context (thereby introducing elements of structure e.g. systems of communication and authority). However, and given the observed organisational bias of STIN [Meyer, 2007], we feel that the influence of an organisation's external environment may have initially been downplayed by its adoption. As discussed in Section 7.4.4, when revisiting the conceptual model, analysis of the data showed the impact of an organisation's external environment on its security V&V practice. Thus we recall the noted difficultly of identifying what belongs within

a network [Kling et al., 2003, Meyer, 2007] ("[t]he STIN analyst does not open up all network nodes recursively; some must necessarily be left unexamined if any conclusions are ever to be drawn!" [Kling et al., 2003, p. 56]). Whilst this is a universal problem (i.e. regardless of the adopted framework), researchers must consider their data carefully and react accordingly since, in our case, it transpired that the external environment was a stronger influence on the tasks being performed within an organisation than initially expected. Similarly, Suri initially felt that the use of STIN would have led them "towards some pre-conceived notions" [Suri, 2013, p. 291], which may have negatively impacted their research, however, they indicate that their resulting analysis confirmed that this did not happen i.e. the data obtained was not limited by the themes within STIN.

Although the prescriptive nature of STIN has previously been acknowledged [Meyer, 2006, Meyer, 2007], but which has also been viewed as a benefit [Reinert, 2009], we encountered, a number of times during our analysis, a sense of 'does this bit of analysis go here, rather than there'. Namely, we encountered a sense of pigeonholing. For example, a discussion on undesired interactions could involve a resource flow, and resource flows could lead into a discussion concerning communication (which, in terms of STIN, is a step before resource flows). Similarly, how does one structure a discussion on the communication tools which exist within a socio-technical network i.e. what takes precedence? Should the tools themselves be discussed in the context of non-human interactors (thus forming part of a study's interactor analysis), or should they be discussed under the network's existing communication forums (thereby comprising part of the analysis of a network's relationships)? Or should they come under both? Specifically, how does one present, and structure, a coherent narrative based around the key themes of STIN without introducing redundancy in the analysis or, in turn, potentially overlooking aspects of an analysis? This has not been adequately addressed for all study contexts, and we observe that whilst several STIN-based studies successfully structure their analyses across the key themes of STIN e.g. [Meyer, 2007, Reinert, 2009], others follow a less prescriptive approach e.g. [Crowston, 2015, Taylor-Smith, 2016] (although both map sections of their analyses to the STIN themes).

Whilst the use of STIN did not preclude the presenting of our analysis in another form i.e. we did not have to structure our discussion by STIN's individual themes, we welcomed the structure afforded by STIN. For example, mapping questions to specific STIN themes ([Taylor-Smith, 2016] adopts a similar approach), assisted the initial process of data analysis by helping categorise findings by theme. Given the amount of data obtained - across both phases of data collection - a means of structuring it for analysis was required. Without this structure we feel that we would have soon been

"drowning in a sea of data" [Rowley, 2012, p. 263] (especially within the early stages of analysis). However, more often than not, when analysing the data structured by theme, we found that the findings would sit comfortably within a single theme (and which could then be further decomposed into sub-themes); although, as above, there were several occasions when we had to decide where best a finding should be positioned within our analysis. Ultimately, how best to structure an analysis will become apparent as the data is categorised and a narrative takes shape. Whilst we have observed a variety of approaches to structuring the results of a STIN analysis, we find, like the adoption of STIN itself (see [Kling et al., 2003]), that a flexible approach should be taken to result presentation.

As with others adopting STIN (e.g. [Shachaf and Rosenbaum, 2009]), we also felt that it was necessary to explicitly demark the technical and non-technical stakeholders (e.g. developers and marketing). This forced us to look beyond the technical - thus avoiding a solely technical focus which, in a V&V context for example, helped counter the finding of [Rooksby et al., 2009] concerning the software testing literature more generally i.e. that it is too focused on technical issues. Further, and in addition to the type of interactors, we also felt that it was important to actively distinguish between an interactors' level of engagement with a task (in this case, a V&V activity). Whilst [Letch and Carroll, 2007] attempted this visually, we opted to categorise groups of interactors as primary, secondary and tertiary interactors to help show their level of engagement with the activities. Although this proved adequate in this specific context, we feel, in turn, that future studies employing STIN should not just reflect upon the identified interactors' relationships with the underlying task, in the form of their engagement (like [Letch and Carroll, 2007]), but that they should also actively include, and distinguish, between an interactors' level of engagement and their level of influence. Since, as we found, some interactors - especially those within the tertiary group - although indirectly involved with the V&V activities (thereby showing limited engagement with a task), actually had considerably more influence over the activities than many of the other interactors who were more directly engaged with the activities (e.g. see Section 6.3.8.8). We feel this is an important distinction - especially when attempting to visualise a STIN.

However, whilst STIN models can be used to further help explain the relationships within a socio-technical network we, like [van der Merwe, 2010], found that the resulting complexity was too difficult to visualise, and thus elected to forgo their use (others have also questioned their value e.g. [Walker and Creanor, 2009]). Therefore, whilst we did not utilise them in terms of communicating our analysis, we did utilise them when constructing our analysis e.g. after an interview was held, we attempted to visualise

the V&V socio-technical network as found within that organisation. Although often fragmentary, they assisted in the preparation of our analysis (thereby echoing [Walker and Creanor, 2009]), rather than in presenting a complete analysis. This is potentially attributable to the inherent complexity of the topic under study (i.e. the encountered diversity in terms of both human and non-human interactors, as well as their relationships), thus the visualisation would likely have ended up obscuring, rather than clarifying and presenting, the overall situation. However, this appears dependent on context: whilst [Meyer, 2007] did not engage with such visualisations, [Taylor-Smith, 2016] did. Specifically, Taylor-Smith presented STIN models at a level of analysis comparable to our focus on a single organisation. In this instance, the models provided a concise overview of the interactors, their motivations, the excluded actors and undesired interactions and resource flows. Whilst this proved useful, and feasible, for the three case studies conducted by Taylor-Smith, we recall our intent to understand security V&V practice within software SMEs more generally, therefore, presenting a series of models per organisation would not have helped in our quest of being able to generalise our findings concerning such practice within our context. Thus, a researcher employing STIN should consider the communication intent of the STIN models, namely, either in terms of highlighting a single case study, or in attempting to generalise a series of findings overall (with the acknowledgment that, dependent on complexity, the model may soon become unwieldy, thereby defeating the purpose of the visualisation). As noted by Walker and Creanor: "different levels of resolution will be appropriate to different analyses" [Walker and Creanor, 2009, p. 306].

Although we acknowledged the nebulous nature of STIN [Meyer, 2007], as well as its continued uptake (see Section 3.3.6), we recall that many variations exist as to how best to study the relationship between the social and the technical [Aidemark, 2007]. Therefore, and in addition to STIN, we also examined ANT and SCOT (which is also where STIN draws influence, see Section 3.3). Whilst all of these offer the opportunity for studying technology, we observe that ANT and SCOT "are more commonly used to understand how new technologies are socially constructed during their creation, and relatively infrequently focus on their subsequent regular use" [Meyer, 2007, p. 262] (similar is echoed in [Orlikowski and Iacono, 2001]). We recall that one of the fundamental assumptions underlying STIN is that a focus on routine use is critical [Kling et al., 2003]. Meyer provides the example that whilst marine mammal scientists use photo-identification, they do not create the digital cameras involved. Similarly, many of the tools and technologies we found being utilised by developers and test engineers, so as to perform security V&V, were not created by them. Further, as our study context was effectively an empirical unknown (see Section 1.3.6 and [Kreeger

and Harindranath, 2012]), and that people within small organisations "are always busy with their daily routines" [Wong and Aspinwall, 2004, p. 54], understanding routine use, within a normal, everyday setting [Meyer, 2007], has obvious value.

This focus on the routine, and the ordinary, also helped provide a more grounded, and complete, analysis. Further, we highlight, like others (e.g. [Meyer, 2007, Suri, 2013, Morrison, 2014, Taylor-Smith, 2016]), the value in STIN's surfacing, and bringing into focus, excluded actors and undesired interactions (Kling was known for making "the unobvious, the taken-for-granted, and the ignored explicit, problematic, and visible" [Robbin, 2007, p. 245]). However, whilst it is difficult to state with certainty that the adoption of either ANT or SCOT would not have identified these aspects, we feel that the chances of their doing so - in comparison with STIN - would have been reduced. Specifically, in terms of SCOT, we find that the focus on 'relevant social groups' can lead to groups who have no voice, or have been suppressed - but who will, nonetheless, be impacted - becoming overlooked [Winner, 1993]. As Meyer states: "[t]his bias results in telling the stories of technology from the point of view of powerful players at the expense of the less powerful and the excluded" [Meyer, 2007, p. 44] (a similar focus on the prominent and the powerful has been identified in terms of ANT [McLean and Hassard, 2004, Mitev, 2009, Mwenya and Brown, 2017]). We recall, for example, that test engineers, and test-based activities, have been considered second-class [Bertolino, 2003, Juristo et al., 2006]. Thus, unlike ANT, it would have been unwise to have considered all actors as being equal [Walsham, 1997, Mitev, 2009, Sayes, 2017]. Therefore, by including questions which helped surface excluded actors (in our case, in terms of interactors participating in individual V&V activities), we were able to obtain a more complete picture (as [Meyer, 2014] found, such questions helped uncover richer information). However, rather than just focusing on excluded actors in their entirety, we found it useful to differentiate between desired and undesired actor exclusions (see Section 5.6.5). As we found, some actors were excluded, but should not have been; whereas, in some instances, it was felt that actors should be excluded. This distinction has further implications on an interactor's relationships with other interactors in the socio-technical network (these can then be explored under the associated undesired interactions).

In summary, we find STIN well suited to the study of V&V (although there is evidence of its uptake - across a variety of areas - we find that its application to software engineering is limited to a few existing studies e.g. [Scacchi, 2005, Amrit, 2008, Jensen, 2010]). Specifically, we found its focus on undesired interactions appropriate for studying a set of activities known to involve conflict between the primary interactors involved (e.g. [Sawyer, 2001, Cohen et al., 2004, Juristo et al., 2006, Vogel, 2011, Zhang

et al., 2014]). Further, we were able to successfully conceptualise V&V as a STIN, showing the importance of striking equilibrium between the social and technical realities surrounding such practice. Notably, reinforcing that if we are to improve practice, it is necessary to focus on both the social and technical aspects (e.g. organisations cannot simply invest in new tools or the adoption of new methodologies without being cognisant of the surrounding social factors, see Section 8.3.2.1). Our findings, through the application of STIN, help confirm the complex socio-technical nature of V&V. However, whilst STIN, overall, contributed and imparted value when applied to our specific study context, we also acknowledge that although time has elapsed since it was considered as being somewhat nebulous (see [Meyer, 2007]), we find there is still limited discussion concerning its adoption and application. Specifically, like [Taylor-Smith, 2016], we had to review a variety of existing STIN-based studies to determine how STIN had been employed, and within what contexts, in order to increase our understanding of how STIN could be operationalised. Our study contributes to the STIN literature by identifying another 'problem space' suited to its application, as well as by demonstrating the research methods which could be employed within such a context (thereby helping address some of the challenges identified within [Meyer, 2006], see Sections 8.2.1 and 8.2.3). Notably, when undertaking future empirical software engineering research (e.g. see Section 8.4), we would continue to elect to employ STIN.

8.3.2 Implications for Practice

Within this section we present several implications for practice. We structure these resulting implications across three different groups of interactors: an organisation's management, those performing the V&V activities (i.e. the practitioners), as well as those in academia (who can also contribute towards improving practice).

8.3.2.1 Management

Given the importance of an organisation's management being seen to support the V&V activities (whether in terms of enforcing processes [Testa, 2009], or ensuring that test engineers remain motivated [Perry, 2006]), we recall our findings showing that management is not a strong motivator in this regard (see Sections 5.6.4.2, 5.6.5.2 and 6.3.5.3). Therefore, it is clear that an organisation's management must establish a greater presence in terms of these activities. In particular, engineering management (e.g. a head of engineering) should visibly be seen to support the improvement and practice of security V&V within an organisation. We find that this is most likely to gather momentum if they are directly engaged with the activities. However, in terms

of an organisation's management in general (e.g. a CEO or CTO), and acknowledging that they are less likely to have the time to directly engage with the activities; they, at the very least, should be seen to endorse the activities, thereby helping gather traction (assigning this as a formal objective to the head of engineering, or equivalent, would further help cement this). Specifically, one of the primary changes management could make is the appointment of a security V&V champion (we discuss this further in Section 8.3.2.2). However, it is not just the appointment of such an individual where management can help, it is in bestowing them with the authority necessary to drive change throughout an organisation (this could further be reinforced, for example, by their reporting (potentially just a dotted line) into engineering management).

Additionally, and acknowledging the reluctance of developers in performing any form of test-based activity, management should ensure that all developer-focused role descriptions, and responsibilities, include an element of testing. This is clearly necessary given the continued focus of developers on code and review-based activities (this was observed within organisations regardless of their adoption of Agile, see Sections 5.6.1 and 6.3.1, i.e. we find that the traditional role divide between developers and test engineers continues to persist), as well as how developers view test-based activities (see Section 6.3.6.2). Further, we recall that developers are probably the most suited of the primary interactors to perform security-focused testing given their intimate product knowledge (effective security testing requires such a level of knowledge [Thompson et al., 2002, Potter and McGraw, 2004]) - a view echoed by [Rindell et al., 2015] - however, the obvious benefits of this knowledge will remain unused unless developers are either given the time or the incentive to perform these activities. Effectively, an organisation's management should actively help instil a degree of respect for the test-based activities (e.g. by not immediately treating the activities as dispensable as we, and others, have observed e.g. [Rodrigues et al., 2010], see Section 5.6.5.2). Notably, when developers become responsible for testing, their levels of test-awareness increase [Talby et al., 2006] which, in turn, is likely to improve their perception of test-based activities. Similarly, test engineers should be encouraged, and developed within an organisation, so that they can embrace review-based activities (we recall the value of code reviews as a means of communicating key product information [Coram and Bohner, 2005]). By targetting both primary interactors, and attempting to bring them closer together in terms of the activities they perform, not only is a stronger relationship likely to result between them, but they will become more effective in terms of performing V&V.

Further, whilst we found that Agile is practiced somewhat inconsistently within organisations, we feel that the Agile concept of pairing engineers (in this case a developer and test engineer on a security V&V task) could offer a potential solution,

or compromise, to the observed traditional role divide e.g. the test engineer benefits from the developer's product knowledge, the developer benefits from the test engineer's knowledge on how to construct tests and how to break software. However, this requires a strong working relationship between the two types of interactor - although we recall that their relationship is fraught with difficulties (see Section 6.3.6.3) - therefore, this comes back to an organisation's management helping ensure that test-based activities, and test engineers, are not viewed negatively (e.g. that they are no longer perceived as being second-class [Bertolino, 2003, Juristo et al., 2006]). Notably, and given the inconsistency in Agile adoption (even within the context of a single organisation), organisations should strive for a degree of consistency in practice, since this is likely to improve the relationship between developers and test engineers through the transparency of tasks, common objectives (at a scrum team level), as well as through frequent communication between interactors (thereby helping address the identified awareness issues, see Section 7.3.2). Given the differing skill sets and mindsets between the primary interactiors [Pettichord, 2000, Sawyer, 2001, Cohen et al., 2004, Dhaliwal et al., 2011, Zhang et al., 2014, Zhang et al., 2018], it should be considered imperative for an organisation's management to seek every opportunity to strengthen the relationship between the primary interactors since, a failure to do so, will ultimately impact product quality [Zhang et al., 2018].

Finally, and given the impact of an organisation's information security culture on security V&V practice (see Sections 5.8.2 and 6.4.2), organisations should continue to strengthen their information security culture. This is likely to be the most optimal way of instilling a security-focused mindset into the non-technical interactors which exist within the V&V socio-technical network (who, we recall, can significantly impact the V&V activities, particularly in terms of resource flows, see Section 6.3.6.5).

8.3.2.2 Practitioners

We discussed above the nominating of a security V&V champion - this individual should be (but does not necessarily have to be) someone from within the organisation itself. This is for two reasons. Firstly, from a technical perspective, it ensures that the individual has an appropriate amount of product domain knowledge (others have echoed the importance of this when performing security V&V e.g. [McDermott, 2000]). Secondly, from a social perspective, we imagine that such an individual is likely to be a senior engineer with established tenure within their organisation. This will help establish trust and respect with other employees (which is necessary as the individual will be providing guidance and instruction to multiple teams). This is especially important given our findings, as well as those of others e.g. experienced reviewers are known to

resist adapting their approach when receiving external suggestions [Kelly and Shepard, 2002] and the attitudes of senior engineers can shape those of junior members in terms of test-based activities [Shah and Harrold, 2010]. Thus an organisation should carefully consider who they nominate to such a position.

Significantly, the security V&V champion should focus on helping reduce the observed 'someone else's problem' view when it comes to security V&V (see Section 7.3.2). To be successful in this aim, it will be necessary for this individual to be technically astute enough to engage with both developers and test engineers (and to be able to deal with their differing skill sets and mindsets [Pettichord, 2000, Sawyer, 2001, Dhaliwal et al., 2011, Zhang et al., 2014, Zhang et al., 2018]), as well as non-technical stakeholders. Notably, the nominated individual does not necessarily need to be dedicated in role (i.e. we acknowledge employees wear multiple hats within SMEs [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018] and also recall that the Security Test Engineer [O:RV-I:BM], from within one medium-sized organisation, also performed non-security-focused testing when required). However, as above, such an individual will also need the authority and support of an organisation's management to be able to add value (otherwise there is a danger that they would just become overlooked and become viewed as no more than a nuisance). Additionally, whilst we have discussed nominating a champion, within some organisations it might be necessary to nominate more than one. Although our findings support [Beznosov and Kruchten, 2004], namely, that a single security-focused engineer can support multiple teams (thereby suggesting one such individual is sufficient in an SME context), we feel, especially given the different skill sets and mindsets which exist between the primary interactors, and which we encountered, that it might be necessary to nominate a security V&V champion from each discipline (i.e. development and test). To a large extent, this is dependent on the individuals within an organisation (whilst [Rindell et al., 2015] indicate that developers are the most appropriate interactor to assume such a security specialist role, we found evidence showing that a test engineer could just as well fulfil the role of security V&V champion).

Further acknowledging the importance of a security V&V champion, we find that such an individual can directly benefit others within an organisation. In particular, and observing the limited training afforded within software SMEs, this individual, when training opportunities do become available, can help propagate this amongst the wider team (as we found, see Section 5.6.7.3, mentoring was one of the primary forms of training encountered). This reflects the view of others in terms of spreading security engineering knowledge more generally e.g. [Wäyrynen et al., 2004]. However, and regardless of whether a security V&V champion is nominated, and given the ever-

evolving nature of security [Khan and Parkinson, 2018], organisations should actively ensure that their knowledge remains up to date (as found, the majority of training received, regardless of mechanism, occurred over 12 months ago). We recall that one of the easiest ways to improve security is to learn from our mistakes [Pfleeger and Pfleeger, 2012]; given the tendency to continually repeat the same security-related mistakes [Piessens, 2002, McGraw, 2008, Anderson, 2008], demonstrates that organisations need to increase their level of security V&V knowledge within the organisation itself. This is important, since whilst a level of reliance on outsourced engineers might, and should be, expected, in order to ensure that the communication between an organisation's engineering team (i.e. it should involve both developers and test engineers) and the outsourced engineers engaged is productive, it is important that the organisation's engineers understand the information being conveyed, as well as how to respond to it (for example, within one organisation we found that the developers did not understand the security risks communicated within the report from the outsourced agency; within another organisation we found that the test engineers were excluded from seeing the resulting report, see Section 6.3.7.4).

Many of our findings also demonstrate the necessity of improving the inclusivity of the security V&V activities. This can be achieved without a security V&V champion, but requires improvement in the communication between the interactors - both human and non-human - within the V&V socio-technical network. For example, and given the evident reliance on outsourced engineers for the security V&V activities (see Sections 5.6.2 and 6.3.8.7), we find that widening the communication between the outsourced engineers, and an organisation's engineering team, will not only help improve the associated levels of awareness, and potentially help guide and develop the engineers, it would also help reduce the encountered 'someone else's problem' view. Similarly, but in a non-human context, organisations should ensure that vulnerabilities are appropriately - and consistently - labelled within their defect tracking systems. A failure to adopt such practice reduces the ability for an organisation to communicate vulnerability information internally and externally i.e. they can become lost amongst defects [Peters et al., 2017]. However, most notably, it impacts their ability to identify the scope of the security problem within a product, in turn, preventing their ability to identify other vulnerable areas of code (see, for example, [Neuhaus et al., 2007]). Given the limited resources of software SMEs, such information is essential in helping determine where interactors should focus their security V&V efforts.

In summary, improving an organisation's internal security V&V practice will only be possible if the inclusivity of the associated activities improves. This can be assisted by increasing the levels of communication surrounding such practice which, in turn,

will further improve awareness levels. Identifying a security V&V champion to drive this change appears necessary. It is only by becoming less reactive to security that we may start to see a reduction in the number of vulnerabilities being reported - especially those which demonstrate that, as practitioners, we have failed to learn our lessons.

8.3.2.3 Academia

Although we did not explicitly explore the role of the academic within the V&V socio-technical network, it is clear that there are implications, resulting from this study, that are also pertinent to such a group of interactors. Specifically, although we had already noted that there exists a lack of university-based training around the security V&V activities (e.g. [Kreeger, 2009]), given the prominence afforded to security-related failures, and with customers becoming more security aware [Abawajy, 2014, Zabicki and Ellis, 2017], we had expected some level of improvement in this regard. However, neither phase of data collection and analysis either alluded to, or demonstrated, that any security V&V knowledge had been obtained during a respondent's or interviewee's university education. The implication of this is that practitioners do not have a strong base on which to develop their knowledge. Whilst this would not be such a problem if software SMEs actively addressed such knowledge gaps, as we found, software SMEs are unlikely to provide much in the way of training - especially in terms of the security V&V activities (see Section 5.6.7.3).

Therefore, security V&V should become a more emphasised component within university-based software engineering courses. This would also help counter the apparent 'not for me' attitude expressed by developers in terms of test-based activities (see Section 6.3.6.2). Notably, we are not suggesting that a dedicated course is required, but affording V&V equal importance in a university's curricula may help reduce the encountered bias (as observed within Section 2.6, test is an underemphasised element in university curricula). Further, by having a strong base on which to build an employee's security V&V knowledge, employees are also more likely to continue to learn, and develop, through the forms of training which software SMEs appear more comfortable in supporting e.g. through on-the-job and self-directed learning (which, as both we and others found, e.g. [Lethbridge, 2000, Sung and Paynter, 2006], reflects the training patterns observed within software SMEs, see Section 5.6.7.3). Ultimately, by instilling an appreciation for security V&V, throughout a university's software engineering curriculum, will further help increase the associated awareness levels of, and an appreciation for, the activities themselves for when a graduate enters the workforce. Given that the majority of software producing organisations are small, and are developing significant products [Fayad et al., 2000], it is important that we adequately prepare

future employees for the challenges ahead.

Aside from this, and more generally, academics should continue to ensure that the results from their research are accessible to practitioners, as well as that the realities faced by practitioners do not become overlooked [Parnas and Lawford, 2003]. These are more general problems which, ultimately, also require practitioners to become aware of the work of researchers. We recall the importance of improving the level of awareness surrounding security V&V (see Section 7.3.2) - this is applicable to both researchers and practitioners.

8.3.3 Summary of Recommendations

Given the implications discussed above, we distil these into the following set of recommendations:

- An organisation's management should be visibly proactive in their support of security V&V: this should encompass both the development and enforcement of processes, as well as investment in the activities and the interactors performing them. A failure to obtain such support is likely to result in the activities being considered less important, 'someone else's problem' and dispensable.

- The traditional role-divide between developers and test engineers must continue to be eroded: in order to strengthen the working relationship between the primary interactors, it is important that they are encouraged, and developed, so as to be able to perform both test and review-based activities. A stronger working relationship, between the interactors - and the activities - will result in V&V being performed more effectively within an organisation, in turn, improving product quality. Additionally, improving on how Agile is adopted and applied within an organisation may further assist in the building of this relationship.

- Organisations should continue to strengthen their information security culture: it is important, throughout an organisation, to increase employee awareness of security more generally. This will result in a greater appreciation for the need to support, and further mature, an organisation's security V&V practice - especially by those interactors who, although less directly engaged with the activities, hold considerable influence over them.

- Software SMEs should appoint a recognised security V&V champion: such an individual should be technically astute, able to engage with both developers and test engineers, and capable of propagating security V&V knowledge and awareness throughout their organisation. Notably, they must also be empowered -

and seen to be supported by - their organisation's management, so as to be able to drive effective change throughout their organisation, thereby improving the encountered levels of security V&V inclusivity, awareness and support.

- Organisations must invest in security V&V training: given the ever-evolving nature of software security, it is important that employee knowledge remains up to date. This is necessary not only when communicating internally within an organisation, but also externally (for example, when relying on outsourced engineers for performing an organisation's security V&V activities).

- Communication surrounding security V&V must become more inclusive and consistent: this is necessary to further reduce the encountered 'someone else's problem' view and in terms of helping improve the awareness levels surrounding the security V&V activities being performed more generally. This includes communication between human interactors and communication between human and non-human interactors (for example, adopting a consistent approach to vulnerability labelling, within a defect tracking system, increases the ability to identify vulnerable areas of code, thereby helping focus an organisation's security V&V efforts).

- Academics should ensure that security V&V becomes a more emphasised component within university-based software engineering courses: this will help ensure that software SMEs have a stronger base on which to build, and address, knowledge gaps - particularly in the forms with which such organisations appear to be the most comfortable in supporting. Additionally, emphasising these activities at the university-level will further help socialise them, in turn, helping improve the associated awareness levels and the encountered bias exhibited by the various interactors towards specific V&V activities.

- Security V&V should be considered a socio-technical discipline, with its research addressing the needs of practitioners: security V&V research should address, and reflect, the socio-technical realities faced by practitioners. Therefore, a socio-technical approach, employing both quantitative and qualitative research methods, should be taken to its study. Further, it is important to ensure that the results being generated are accessible to practitioners, thereby helping close the gap between V&V research and practice.

We now discuss the limitations of the research performed, as well as some future research directions.

8.4 Limitations and Future Research Directions

All works of this nature suffer from some form of limitation; thus, whilst we justified our survey response rate - specifically, by observing the impact of researching a highly sensitive subject, especially within the elected organisational and technological contexts of this thesis (see Section 5.2) - we also acknowledge that the number of responses, and the response rate attained, although comparable to other empirical software engineering studies (e.g. [Singer et al., 2008, Thörn and Gustafsson, 2008, Thörn, 2010]), could be perceived as being one such limitation. Similarly, and by extension, the number of interviews conducted could also be viewed as a limitation (however, it was within this phase of the study that we aimed to acquire a depth of understanding as opposed to breadth, see Appendix L).

Therefore, and given the importance of study replication, particularly within an empirical software engineering context [Seaman, 1999, Shull et al., 2008], we believe that replicating this study, initially within a different set of organisations, but employing the same survey instrument and interview guide (Appendices H and K respectively), would further help improve the generalisability of the findings obtained (although we recall the diversity of the organisations reached during this study, see Section 5.4.1). Notably, study replication improves reliability (i.e. it would increase the confidence in our results), helps address threats to validity and also enables us to explore the impact of contextual variations [Krein et al., 2014]. For example, replicating the study within a different national context would clearly be beneficial in furthering our understanding of both V&V practice and the influence of an organisation's information security culture on such practice. Since, in terms of the former, we find examples of such replication (e.g. [Dias-Neto et al., 2017] replicate a survey of software testing practice within a different geographical area) and, in terms of information security culture, we find that the national context has often been overlooked [Dojkovski et al., 2007, Dojkovski et al., 2010]. Therefore, across both aspects, we find utility in replicating the study in order to confirm or challenge our current findings. In addition, and whilst we acknowledged throughout this thesis the importance of addressing the research objectives within a software SME context (see Section 1.2), we feel that it would be a useful, and a logical extension, to conduct a similar study within larger software organisations as well. Aside from enabling an interesting comparison to be drawn, and the identification of common problems, as well as potential solutions, it would also further help reinforce the findings obtained during this study (including confirming the differences which exist between the two types of organisation in terms of a specific set of practices). Also, and given the greater likelihood of such organisations having geographically dispersed development

teams, this would be an interesting study to conduct so as to understand the impact of national cultures mixing together on a shared activity, namely, that of security V&V. Additionally, replicating the study within the same set of organisations, but following the passage of some time, would allow us to gauge the level of change. For example, whether things have improved or declined and the reasons behind this change.

Further, and in terms of corroboration and triangulation at the organisational level, as opposed to the study level (where the findings are corroborated and triangulated across the survey, the interviews and the existing literature), it would be desirable, when replicating, to interview at least two individuals, preferable three, from within each organisation studied. Thus, and given the focus of any future study, this might encompass both a developer and a test engineer (especially if wanting to further explore their relationship in order to obtain 'both sides of the picture'), as well as ensuring that a non-technical stakeholder is interviewed (thereby helping obtain a broader organisational view). Such interactor diversity is important since, as an example, whilst we were able to confirm the view in which developers hold test engineers and test-based activities (see Sections 6.3.6.2 and 6.3.6.3), we also found evidence showing a similar view is harboured by other interactors within an organisation (see Section 6.3.6.4). In particular, we found that non-technical stakeholders also exhibited this view. Given the observed impact by interactors outside of engineering (see Section 6.3.6.5), and the importance of test engineers remaining motivated and supported in their activities, it becomes necessary to fully explore these relationships by also directly engaging with these interactors.

Additionally, whilst we did not explicitly limit, or exclude, any factor from the process of data collection and analysis, given the nature of the study, as well as the research questions themselves, we found that such an approach inherently led to some factors receiving greater treatment than others. However, we readily acknowledge the value of increasing the focus given to some of these lesser examined factors. For example, although the concept of trust was touched upon (e.g. in terms of the reliance being placed on outsourced engineers, see Section 6.3.6.7), we recognise that additional avenues for further exploration exist e.g. by specifically examining the developer-test engineer relationship in order to understand how the levels of trust, between the primary V&V interactors, become cultivated and/or degraded over time. Similarly, whilst we explored aspects of process improvement, we found no evidence to show that the interactors displayed a resistance to such organisational initiatives. Notably, a specific focus on resistance to change would further help support, or challenge, other V&V-focused studies that have previously alluded to its impact e.g. [Kelly and Shepard, 2002]. In summary, our analysis, and thus the resultant focus of the thesis, was driven

by the data itself.

Finally, and aside from replicating the study and varying the contexts, a number of additional avenues to explore surfaced during the course of the research, which we believe warrant further study. For example, it is apparent, even given the evident popularity of Agile (e.g. [Clarke and O'Connor, 2013, Ryan and O'Connor, 2013, Rindell et al., 2015, Chóliz et al., 2015, Rindell et al., 2017], see Figure 5.4), that its relationship with security V&V needs to be further explored (a similar call has been expressed in terms of security more generally e.g. [Rindell et al., 2017]). However, a specific area concerning Agile, which we are keen to pursue, is based on our finding that product owners appear to have limited, direct engagement with any V&V activity (see Sections 5.6.2 and 6.3.2). Given the importance of their role, and their ability to influence and shape the focus of their team, it would be sensible to explore their influence further, in particular, to determine whether their background shapes the software and security V&V focus within their team e.g. although we, and others, have observed that developers typically take on this role within an organisation (e.g. [Diebold et al., 2015, Jovanović et al., 2017], see Section 6.3.2), do we find their preference for review-based activities (see Section 5.6.1), and their view of test-based activities (see Section 6.3.6.2), being exhibited when performing this role? Similarly, and like in [Shah and Harrold, 2010], we found evidence to suggest that senior engineers can impact the attitudes of junior engineers (see Section 6.3.6.3), we feel that this is an important aspect to further explore in terms of cultivating security V&V practice within an organisation.

8.5 Personal Reflections

Although the author of this thesis is a practising software engineer, it is important to observe, in order to demonstrate a reduced subjective bias - especially since reference has been made to previous experiences - that the author, since embarking on this research, has worked for a large organisation and predominately in a role straddling both developers and test engineers. Therefore, over time, an appreciation for both roles has been cultivated. This was important to help prevent any pre-disposition to a particular role surfacing during the process of data collection and analysis. However, it should also be noted that this experience not only assisted with the construction of the survey instrument and the interview guide, but that it also helped immeasurably in terms of building rapport with the interviewees (as evidenced by their openness and honesty). This relationship may have otherwise been unattainable without the shared knowledge and understanding which existed between researcher and interviewee.

Further, and although straddling both industry and academia, we were genuinely

surprised to find the gap, between the state-of-the-art in terms of research, and that concerning practice, being exhibited on such a wide scale. We were also surprised, more generally, by the lack of security V&V responsibility, knowledge, and awareness, which was found to exist within the organisations. If nothing else is taken from this thesis, we would hope that the 'call to arms' is taken to heart in terms of software SMEs actively improving their security V&V capability. Given our reliance on software, much of which originating from within SMEs, we need to ensure they are conscious of the need to improve and are given guidance on how best to achieve this. As academics, it is in our interest to ensure that the tools, technologies and methodologies we develop are both accessible and useful to practitioners i.e. that they take into account the surrounding socio-technical realities and solve real-world problems. Therefore, and given the importance of software, SMEs, and security, we intend to continue to explore their intersection; initially by building upon the knowledge already obtained, and through addressing, and exploring, the areas identified in Section 8.4.

We look forward to continuing this journey.

Bibliography

[Aaltonen and Kujala, 2016] Aaltonen, K. and Kujala, J. (2016). Towards an improved understanding of project stakeholder landscapes. *International Journal of Project Management*, 34(8):1537–1552.

[Abawajy, 2014] Abawajy, J. (2014). User preference of cyber security awareness delivery methods. *Behaviour & Information Technology*, 33(3):237–248.

[ACM, 2020] ACM (2020). ACM Digital Library. `https://dl.acm.org/`. Retrieved: 16th of January, 2020.

[Acuña et al., 2006] Acuña, S. T., Juristo, N., and Moreno, A. M. (2006). Emphasizing Human Capabilities in Software Development. *IEEE Software*, 23(2):94–101.

[Adrion et al., 1982] Adrion, W. R., Branstad, M. A., and Cherniavsky, J. C. (1982). Validation, Verification, and Testing of Computer Software. *Computing Surveys*, 14(2):159–192.

[Ågerfalk, 2013] Ågerfalk, P. J. (2013). Embracing diversity through mixed methods research. *European Journal of Information Systems*, 22(3):251–256.

[Ahmed, 2009] Ahmed, A. (2009). *Software Testing as a Service*. Auerbach Publications.

[Ahmed, 2011] Ahmed, A. (2011). *Software Project Management: A Process-Driven Approach*. Auerbach Publications.

[Ahrens and Dent, 1998] Ahrens, T. and Dent, J. F. (1998). Accounting and Organizations: Realizing the Richness of Field Research. *Journal of Management Accounting Research*, 10:1–39.

[Aidemark, 2007] Aidemark, J. (2007). IS Planning and Socio-technical Theory Perspectives. In *EMCIS '07: Proceedings of the European and Mediterranean Conference on Information Systems*.

[Alcadipani and Hassard, 2010] Alcadipani, R. and Hassard, J. (2010). Actor-Network Theory, organizations and critique: towards a politics of organizing. *Organization*, 17(4):419–435.

[Allen et al., 2008] Allen, J. H., Barnum, S., Ellison, R. J., McGraw, G., and Mead, N. R. (2008). *Software Security Engineering: A Guide for Project Managers*. Pearson Education.

[Allison, 2010] Allison, I. (2010). Organizational Factors Shaping Software Process Improvement in Small-Medium Sized Software Teams: A Multi-Case Analysis. In *QUATIC '10: Proceedings of the 7th International Conference on the Quality of Information and Communications Technology*, pages 418–423. IEEE Computer Society.

[Allison and Merali, 2007] Allison, I. and Merali, Y. (2007). Software process improvement as emergent change: A structurational analysis. *Information and Software Technology*, 49(6):668–681.

[Alter, 2012] Alter, S. (2012). Exploring the Temporal Nature of Sociomateriality from a Work System Perspective. In *AMCIS '12: Proceedings of the Eighteenth Americas Conference on Information Systems*, number 13.

[Ammann and Offutt, 2008] Ammann, P. and Offutt, J. (2008). *Introduction to Software Testing*. Cambridge University Press.

[Ammenwerth, 2010] Ammenwerth, E. (2010). Evidence Based Health Informatics. In Hovenga, E. J. S., editor, *Health Informatics: An Overview*, pages 427–434. IOS Press.

[Amrit, 2005] Amrit, C. (2005). Coordination in Software Development: The Problem of Task Allocation. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7.

[Amrit, 2008] Amrit, C. (2008). *Improving Coordination in Software Development through Social and Technical Network Analysis*. PhD thesis, University of Twente.

[Anderson, 2001] Anderson, R. (2001). Why Information Security is Hard - An Economic Perspective. In *ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference*, pages 358–365. IEEE Computer Society.

[Anderson and Moore, 2006] Anderson, R. and Moore, T. (2006). The Economics of Information Security. *Science*, 314(5799):610–613.

[Anderson, 2008] Anderson, R. J. (2008). *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, second edition.

[Andersson and Runeson, 2002] Andersson, C. and Runeson, P. (2002). Verification and Validation in Industry - A Qualitative Survey on the State of Practice. In *ISESE '02: Proceedings of the 2002 International Symposium on Empirical Software Engineering*, pages 37–47. IEEE Computer Society.

[Antony et al., 2005] Antony, J., Kumar, M., and Madu, C. N. (2005). Six sigma in small- and medium-sized UK manufacturing enterprises: Some empirical observations. *International Journal of Quality & Reliability Management*, 22(8):860–874.

[Antony et al., 2016] Antony, J., Vinodh, S., and Gijo, E. V. (2016). *Lean Six Sigma for Small and Medium Sized Enterprises: A Practical Guide*. CRC Press.

[Appelbaum, 2001] Appelbaum, E. (2001). Transformation of work and employment and new insecurities. In *The Future of Work, Employment and Social Protection: The search for new securities in a world of growing uncertainties*, pages 17–37. International Institute for Labour Studies.

[Araújo et al., 2013] Araújo, A. F., Rodrigues, C. L., Vincenzi, A. M. R., Camilo, C. G., and Silva, A. F. (2013). A Framework for Maturity Assessment in Software Testing for Small and Medium-Sized Enterprises. In *SERP '13: Proceedings of the 2013 International Conference on Software Engineering Research and Practice*.

[Ardis and Mead, 2011] Ardis, M. and Mead, N. R. (2011). The Development of a Graduate Curriculum for Software Assurance. In *AMCIS '11: Proceedings of the Seventeenth Americas Conference on Information Systems*, number 34.

[Arkin et al., 2005] Arkin, B., Stender, S., and McGraw, G. (2005). Software Penetration Testing. *IEEE Security & Privacy*, 3(1):84–87.

[Arora et al., 2006] Arora, A., Caulkins, J. P., and Telang, R. (2006). Research Note: Sell First, Fix Later: Impact of Patching on Software Quality. *Management Science*, 52(3):465–471.

[Arora et al., 2008] Arora, A., Telang, R., and Xu, H. (2008). Optimal Policy for Software Vulnerability Disclosure. *Management Science*, 54(4):642–656.

[Asghari et al., 2016] Asghari, H., van Eeten, M., and Bauer, J. M. (2016). Economics of cybersecurity. In Bauer, J. M. and Latzer, M., editors, *Handbook on the Economics of the Internet*, chapter 13, pages 262–287. Edward Elgar Publishing Ltd.

[Aurum et al., 2002] Aurum, A., Petersson, H., and Wohlin, C. (2002). State-of-the-Art: Software Inspections after 25 Years. *Software Testing, Verification and Reliability*, 12(3):133–154.

[Austin and Williams, 2011] Austin, A. and Williams, L. (2011). One Technique is Not Enough: A Comparison of Vulnerability Discovery Techniques. In *ESEM '11: Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement*, pages 97–106. IEEE Computer Society.

[Ayyagari et al., 2007] Ayyagari, M., Beck, T., and Demirgüç-Kunt, A. (2007). Small and Medium Enterprises across the Globe. *Small Business Economics*, 29(4):415–434.

[Babar et al., 2007] Babar, M. A., Verner, J. M., and Nguyen, P. T. (2007). Establishing and maintaining trust in software outsourcing relationships: An empirical investigation. *Journal of Systems and Software*, 80(9):1438–1449.

[Baca and Carlsson, 2011] Baca, D. and Carlsson, B. (2011). Agile Development with Security Engineering Activities. In *ICSSP '11: Proceedings of the 2011 International Conference on Software and Systems Process*, pages 149–158. ACM Press.

[Bach, 1997] Bach, J. (1997). Test Automation Snake Oil. In *Proceedings of the 14th International Conference and Exposition on Testing Computer Software.*

[Baharom et al., 2005] Baharom, F., Deraman, A., and Hamdan, A. R. (2005). A Survey on the Current Practices of Software Development Process in Malaysia. *Journal of Information and Communication Technology*, 4:57–76.

[Bajaj and Balram, 2009] Bajaj, S. K. and Balram, S. (2009). Incorporating Software Testing as a Discipline in Curriculum of Computing Courses. In Yang, J., Ginige, A., Mayr, H. C., and Kutsche, R.-D., editors, *Information Systems: Modeling, Development, and Integration*, volume 20 of *Lecture Notes in Business Information Processing*, pages 404–410. Springer.

[Baker, Jr., 1997] Baker, Jr., R. A. (1997). Code Reviews Enhance Software Quality. In *ICSE '97: Proceedings of the 19th International Conference on Software Engineering*, pages 570–571. ACM Press.

[Balci et al., 2002] Balci, O., Nance, R. E., Arthur, J. D., and Ormsby, W. F. (2002). Expanding Our Horizons in Verification, Validation, and Accreditation Research and Practice. In *WSC '02: Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers*, pages 653–663. Winter Simulation Conference.

[Banker et al., 1989] Banker, R. D., Datar, S. M., and Zweig, D. (1989). Software Complexity and Maintainability. In *ICIS '89: Proceedings of the 10th International Conference on Information Systems*, pages 247–255. ACM Press.

[Barab et al., 2003] Barab, S. A., MaKinster, J. G., and Scheckler, R. (2003). Designing System Dualities: Characterizing a Web-Supported Professional Development Community. *The Information Society*, 19(3):237–256.

[Barry, 2002] Barry, N. (2002). The Stakeholder Concept of Corporate Control Is Illogical and Impractical. *The Independent Review*, 6(4):541–554.

[Bashir and Goel, 1999] Bashir, I. and Goel, A. L. (1999). *Testing Object-Oriented Software: Life Cycle Solutions*. Springer.

[Basili, 2006] Basili, V. R. (2006). Is There a Future for Empirical Software Engineering? In *ISESE '06: Proceedings of the 5th ACM/IEEE International Symposium on Empirical Software Engineering*, pages 1–1. ACM Press.

[Basili et al., 1996] Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., and Zelkowitz, M. V. (1996). The Empirical Investigation of Perspective-Based Reading. *Empirical Software Engineering*, 1(2):133–164.

[Basili and Selby, 1987] Basili, V. R. and Selby, R. W. (1987). Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*, 13(12):1278–1296.

[Basri and O'Connor, 2010] Basri, S. and O'Connor, R. V. (2010). Understanding the Perception of Very Small Software Companies towards the Adoption of Process Standards. In Riel, A., O'Connor, R., Tichkiewitch, S., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, volume 99, pages 153–164. Springer.

[Baxter, 2000] Baxter, L. F. (2000). Bugged: the Software Development Process. In Prichard, C., Hull, R., Chumer, M., and Willmott, H., editors, *Managing Knowledge: Critical Investigations of Work and Learning*, chapter 3, pages 37–48. Macmillan.

[Bayrak, 2013] Bayrak, T. (2013). A decision framework for SME Information Technology (IT) managers: Factors for evaluating whether to outsource internal applications to Application Service Providers. *Technology in Society*, 35(1):14–21.

[Beck, 1998] Beck, K. (1998). Extreme Programming: A Humanistic Discipline of Software Development. In Astesiano, E., editor, *Fundamental Approaches to Software Engineering*, volume 1382 of *Lecture Notes in Computer Science*, pages 1–6. Springer.

[Beecham et al., 2008] Beecham, S., Baddoo, N., Hall, T., Robinson, H., and Sharp, H. (2008). Motivation in Software Engineering: A systematic literature review. *Information and Software Technology*, 50(9-10):860–878.

[Beekhuyzen et al., 2012] Beekhuyzen, J., von Hellens, L., and Nielsen, S. (2012). Insights from the Underground: Using ANT to Understand Practices and Motivations for File Sharing in Online Communities. In *ECIS '12: Proceedings of the 20th European Conference on Information Systems*, number 49.

[Begel and Nagappan, 2007] Begel, A. and Nagappan, N. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 255–264. IEEE Computer Society.

[Berling and Runeson, 2003] Berling, T. and Runeson, P. (2003). Evaluation of a Perspective Based Review Method Applied in an Industrial Setting. *IEE proceedings Software*, 150(3):177–184.

[Berner et al., 2005] Berner, S., Weber, R., and Keller, R. K. (2005). Observations and Lessons Learned from Automated Testing. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pages 571–579. ACM Press.

[Bernstein and Turban, 2018] Bernstein, E. S. and Turban, S. (2018). The impact of the 'open' workspace on human collaboration. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 373(1753).

[Berry, 2007] Berry, A. (2007). The Importance of SMEs in the Economy. In *International Tax Dialogue (ITD) Global Conference on Taxation of Small and Medium Enterprises*.

[Bertolino, 2003] Bertolino, A. (2003). Software Testing Research and Practice. In *ASM '03: Proceedings of the Abstract State Machines 10th International Conference on Advances in Theory and Practice*, pages 1–21. Springer.

[Beznosov and Kruchten, 2004] Beznosov, K. and Kruchten, P. (2004). Towards Agile Security Assurance. In *NSPW '04: Proceedings of the 2004 New Security Paradigms Workshop*, pages 47–54. ACM Press.

[BIAC and ICC, 2003] BIAC and ICC (2003). Information Security Assurance for Executives: An International Business Commentary on the 2002 OECD Guidelines for the Security of Networks and Information Systems: Towards a Culture of Security. `http://biac.org/wp-content/uploads/2014/05/Final_Information_Security_for_Executives071003.pdf`. Retrieved: 16th of January, 2020.

[BIAC and ICC, 2004] BIAC and ICC (2004). Securing your business: An companion for small or entrepreneurial companies to the 2002 OECD Guidelines for

the security of networks and information systems: Towards a culture of security. `http://biac.org/wp-content/uploads/2014/05/biac-icc_security_for_smes.pdf`. Retrieved: 16th of January, 2020.

[Biehl et al., 2007] Biehl, J. T., Czerwinski, M., Smith, G., and Robertson, G. G. (2007). FASTDash: A Visual Dashboard for Fostering Awareness in Software Teams. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1313–1322. ACM Press.

[Biesenthal and Wilden, 2014] Biesenthal, C. and Wilden, R. (2014). Multi-level project governance: Trends and opportunities. *International Journal of Project Management*, 32(8):1291–1308.

[Bijker, 1997] Bijker, W. E. (1997). *Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change*. MIT Press.

[Bijker, 2009] Bijker, W. E. (2009). Social Construction of Technology. In Olsen, J. K. B., Pedersen, S. A., and Hendricks, V. F., editors, *A Companion to the Philosophy of Technology*, chapter 15, pages 88–94. John Wiley & Sons.

[Bijker, 2010] Bijker, W. E. (2010). How is technology made? - That is the question! *Cambridge Journal of Economics*, 34(1):63–76.

[Bingham-Hall, 2017] Bingham-Hall, J. (2017). *The Blog and the Territory: placing hyperlocal media and its publics in a London neighbourhood*. PhD thesis, University College London.

[Bird et al., 2009] Bird, C., Nagappan, N., Gall, H., Murphy, B., and Devanbu, P. (2009). Putting it All Together: Using Socio-Technical Networks to Predict Failures. In *ISSRE '09: Proceedings of the 20th International Symposium on Software Reliability Engineering*, pages 109–119. IEEE Computer Society.

[Birolini, 2017] Birolini, A. (2017). *Reliability Engineering: Theory and Practice*. Springer, eighth edition.

[Blackwell, 2014] Blackwell, C. (2014). Towards a Penetration Testing Framework Using Attack Patterns. In Blackwell, C. and Zhu, H., editors, *Cyberpatterns: Unifying Design Patterns with Security and Attack Patterns*, chapter 11, pages 135–148. Springer.

[Block, 2011] Block, M. (2011). Evolving to Agile: A story of agile adoption at a small SaaS company. In *Agile '11: Proceedings of the 2011 Agile Conference*, pages 234–239. IEEE Computer Society.

[Boddy, 2016] Boddy, C. R. (2016). Sample size for qualitative research. *Qualitative Market Research: An International Journal*, 19(4):426–432.

[Boehm and Turner, 2005] Boehm, B. and Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, 22(5):30–39.

[Boehm, 1981] Boehm, B. W. (1981). *Software Engineering Economics*. Prentice Hall.

[Boehm and Basili, 2001] Boehm, B. W. and Basili, V. R. (2001). Software Defect Reduction Top 10 List. *IEEE Computer*, 34(1):135–137.

[Bostrom and Heinen, 1977] Bostrom, R. P. and Heinen, J. S. (1977). MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes. *MIS Quarterly*, 1(3):17–32.

[Bradburn et al., 2004] Bradburn, N., Sudman, S., and Wansink, B. (2004). *Asking Questions: The Definitive Guide to Questionnaire Design - For Market Research, Political Polls, and Social and Health Questionnaires*. John Wiley & Sons.

[Brannen, 2005] Brannen, J. (2005). Mixing Methods: The Entry of Qualitative and Quantitative Approaches into the Research Process. *International Journal of Social Research Methodology*, 8(3):173–184.

[Braun and Clarke, 2006] Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101.

[Braun and Clarke, 2013] Braun, V. and Clarke, V. (2013). *Successful Qualitative Research: A Practical Guide for Beginners*. SAGE Publications.

[Brereton et al., 2007] Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., and Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583.

[Briand, 2010] Briand, L. C. (2010). Software Verification - A Scalable, Model-Driven, Empirically Grounded Approach. In Tveito, A., Bruaset, A. M., and Lysne, O., editors, *Simula Research Laboratory*, pages 415–442. Springer.

[Brodman and Johnson, 1994] Brodman, J. G. and Johnson, D. L. (1994). What Small Businesses and Small Organizations Say About the CMM. In *ICSE '94: Proceedings of the 16th International Conference on Software Engineering*, pages 331–340. IEEE Computer Society.

[Brooks, Jr., 1987] Brooks, Jr., F. P. (1987). No silver bullet - essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19.

[Brown and Duguid, 2001] Brown, J. S. and Duguid, P. (2001). Creativity Versus Structure: A Useful Tension. *MIT Sloan Management Review*, 42(4):93–94.

[Bryant and Jary, 1991] Bryant, C. G. A. and Jary, D., editors (1991). *Giddens' Theory of Structuration: A critical appreciation*. Routledge.

[Bryant, 2005] Bryant, S. E. (2005). The Impact of Peer Mentoring on Organizational Knowledge Creation and Sharing: An Empirical Study in a Software Firm. *Group & Organization Management*, 30(3):319–338.

[Bryman, 2006] Bryman, A. (2006). Integrating quantitative and qualitative research: how is it done? *Qualitative Research*, 6(1):97–113.

[Buchwald et al., 2014] Buchwald, A., Urbach, N., and Ahlemann, F. (2014). Business value through controlled IT: toward an integrated model of IT governance success and its impact. *Journal of Information Technology*, 29(2):128–147.

[Budgen and Brereton, 2006] Budgen, D. and Brereton, P. (2006). Performing Systematic Literature Reviews in Software Engineering. In *ICSE '06: Proceedings of the 28th International Conference on Software Engineering*, pages 1051–1052. ACM Press.

[Budnik et al., 2007] Budnik, C. J., Subramanyan, R., and Vieira, M. (2007). Industrial Requirements to Benefit from Test Automation Tools for GUI Testing. In *Lecture Notes in Informatics*, pages 410–414. Gesellschaft für Informatik.

[Bureau van Dijk, 2020] Bureau van Dijk (2020). FAME: The definitive source of companies in the UK and Ireland. `https://www.bvdinfo.com/en-gb/our-products/data/national/fame`. Retrieved: 16th of January, 2020.

[Burge et al., 2008] Burge, J. E., Carroll, J. M., McCall, R., and Mistrík, I. (2008). *Rationale-Based Software Engineering*. Springer.

[Burns et al., 2006] Burns, A., Davies, A., and Beynon-Davies, P. (2006). A study of the uptake of Information Security Policies by small and medium sized businesses in Wales. In *Frontiers of e-Business Research 2006*. Tampere University of Technology and University of Tampere.

[Burnstein, 2003] Burnstein, I. (2003). *Practical Software Testing: A Process-Oriented Approach*. Springer.

[Burton et al., 2011] Burton, S. H., Bodily, P. M., Morris, R. G., Knutson, C. D., and Krein, J. L. (2011). Design Team Perception of Development Team Composition: Implications for Conway's Law. In *RESER '11: Proceedings of the 2nd International Workshop on Replication in Empirical Software Engineering Research*, pages 52–60. IEEE Computer Society.

[Cachia and Millward, 2011] Cachia, M. and Millward, L. (2011). The telephone medium and semi-structured interviews: a complementary fit. *Qualitative Research in Organizations and Management: An International Journal*, 6(3):265–277.

[Calder, 2016] Calder, A. (2016). *Selling Information Security to the Board: A Primer*. IT Governance Publishing, second edition.

[Callon, 1986] Callon, M. (1986). Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay. In Law, J., editor, *Power, Action and Belief: A New Sociology of Knowledge?*, pages 196–223. Routledge & Kegan Paul.

[Camargo et al., 2013] Camargo, K. G., Ferrari, F. C., and Fabbri, S. C. P. F. (2013). Identifying a Subset of TMMi Practices to Establish a Streamlined Software Testing Process. In *SBES '13: Proceedings of the 2013 27th Brazilian Symposium on Software Engineering*, pages 137–146. IEEE Computer Society.

[Camargo et al., 2015] Camargo, K. G., Ferrari, F. C., and Fabbri, S. C. P. F. (2015). Characterising the state of the practice in software testing through a TMMi-based process. *Journal of Software Engineering Research and Development*, 3(1).

[Cao and Ramesh, 2008] Cao, L. and Ramesh, B. (2008). Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*, 25(1):60–67.

[Capgemini et al., 2009] Capgemini, Sogeti, and HP (2009). 2009 World Quality Report. `https://www.uk.sogeti.com/globalassets/uk/reports/sogetiuk_2009-10_world_quality_report.pdf`. Retrieved: 16th of January, 2020.

[Caputo et al., 2016] Caputo, D. D., Pfleeger, S. L., Sasse, M. A., Ammann, P., Offutt, J., and Deng, L. (2016). Barriers to Usable Security? Three Organizational Case Studies. *IEEE Security & Privacy*, 14(5):22–32.

[Carroll et al., 2012] Carroll, N., Richardson, I., and Whelan, E. (2012). Service Science: An Actor-Network Theory Approach. *International Journal of Actor-Network Theory and Technological Innovation*, 4(3):51–69.

[Cater-Steel et al., 2005] Cater-Steel, A., Toleman, M., and Rout, T. (2005). Addressing the Challenges of Replications of Surveys in Software Engineering Research. In *ISESE '05: Proceedings of the 4th ACM/IEEE International Symposium on Empirical Software Engineering*, pages 204–213. IEEE Computer Society.

[Causevic et al., 2009a] Causevic, A., Krasteva, I., Land, R., Sajeev, A., and Sundmark, D. (2009a). A Survey on Industrial Software Engineering. In Abrahamsson, P., Marchesi, M., and Maurer, F., editors, *XP '09: Proceedings of the 10th International Conference on Agile Processes in Software Engineering and Extreme Programming*, pages 240–241. Springer.

[Causevic et al., 2009b] Causevic, A., Krasteva, I., Land, R., Sajeev, A., and Sundmark, D. (2009b). An Industrial Survey on Software Process Practices, Preferences and Methods. Technical report, Mälardalen University MDH-MRTC-233/2009-1-SE.

[Causevic et al., 2010] Causevic, A., Sundmark, D., and Punnekkat, S. (2010). An Industrial Survey on Contemporary Aspects of Software Testing. In *ICST '10: Proceedings of the 2010 Third International Conference on Software Testing, Verification and Validation*, pages 393–401. IEEE Computer Society.

[Cecez-Kecmanovic et al., 2014] Cecez-Kecmanovic, D., Galliers, R. D., Henfridsson, O., Newell, S., and Vidgen, R. (2014). The Sociomateriality of Information Systems: Current Status, Future Directions. *MIS Quarterly*, 38(3):809–830.

[Centeno, 2003] Centeno, C. (2003). Soft Measures to Build Security in e-Commerce Payments and Consumer Trust. *Communications & Strategies*, 51:55–96.

[Chakraborty and Raghuraman, 2015] Chakraborty, P. and Raghuraman, K. (2015). Trends in Information Security. In Information Resources Management Association, editor, *Standards and Standardization: Concepts, Methodologies, Tools, and Applications*, chapter 72, pages 1582–1604. IGI Global.

[Chang et al., 2011] Chang, Y.-Y., Hughes, M., and Hotho, S. (2011). Internal and external antecedents of SMEs' innovation ambidexterity outcomes. *Management Decision*, 49(10):1658–1676.

[Chau and Maurer, 2004] Chau, T. and Maurer, F. (2004). Knowledge Sharing in Agile Software Teams. In Lenski, W., editor, *Logic versus Approximation: Essays Dedicated to Michael M. Richter on the Occasion of his 65th Birthday*, pages 173–183. Springer.

[Chawla et al., 2016] Chawla, P., Chana, I., and Rana, A. (2016). Cloud-based automatic test data generation framework. *Journal of Computer and System Sciences*, 82(5):712–738.

[Cherns, 1976] Cherns, A. (1976). The Principles of Sociotechnical Design. *Human Relations*, 29(8):783–792.

[Chóliz et al., 2015] Chóliz, J., Vilas, J., and Moreira, J. (2015). Independent Security Testing on Agile Software Development: A Case Study in a Software Company. In *ARES '15: Proceedings of the 10th International Conference on Availability, Reliability and Security*, pages 522–531. IEEE Computer Society.

[Chung et al., 2010] Chung, E., Jensen, C., Yatani, K., Kuechler, V., and Truong, K. N. (2010). Sketching and Drawing in the Design of Open Source Software. In *VL/HCC '10: Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 195–202. IEEE Computer Society.

[Ciolkowski et al., 2002] Ciolkowski, M., Laitenberger, O., Rombach, D., Shull, F., and Perry, D. (2002). Software Inspections, Reviews & Walkthroughs. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 641–642. ACM Press.

[Ciolkowski et al., 2003] Ciolkowski, M., Laitenberger, O., Vegas, S., and Biffl, S. (2003). Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering. In Conradi, R. and Wang, A. I., editors, *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, volume 2765, pages 104–128. Springer.

[Clarke and O'Connor, 2013] Clarke, P. and O'Connor, R. V. (2013). An empirical examination of the extent of software process improvement in software SMEs. *Journal of Software: Evolution and Process*, 25(9):981–998.

[Clarke et al., 2010] Clarke, P. J., King, T. M., and Jones, E. L. (2010). WReSTT - Web-Based Repository of Software Testing Tools. In *WTST9: Proceedings of the 9th Annual Workshop on Teaching Software Testing*, pages 52–59.

[Cleary et al., 2014] Cleary, M., Horsfall, J., and Hayter, M. (2014). Data collection and sampling in qualitative research: does size matter? *Journal of Advanced Nursing*, 70(3):473–475.

[CMMI Product Team, 2010] CMMI Product Team (2010). CMMI for Development, Version 1.3. Technical report, Carnegie Mellon University CMU/SEI-2010-TR-033.

[Codenomicon, 2014] Codenomicon (2014). The Heartbleed Bug. http://heartbleed.com/. Retrieved: 16th of January, 2020.

[Coffey and Viega, 2007] Coffey, D. and Viega, J. (2007). Building an Effective Application Security Practice on a Shoestring Budget. https://www.blackhat.com/presentations/bh-usa-07/Coffey_and_Viega/Whitepaper/bh-usa-07-coffey_and_viega-WP.pdf. Retrieved: 16th of January, 2020.

[Cohen et al., 2004] Cohen, C. F., Birkin, S. J., Garfield, M. J., and Webb, H. W. (2004). Managing Conflict in Software Testing. *Communications of the ACM*, 47(1):76–81.

[Coiera, 2007] Coiera, E. (2007). Putting the technical back into socio-technical systems research. *International Journal of Medical Informatics*, 76, Supplement 1:S98–S103.

[Coles-Kemp and Hansen, 2017] Coles-Kemp, L. and Hansen, R. R. (2017). Walking the Line: The Everyday Security Ties that Bind. In Tryfonas, T., editor, *Human Aspects of Information Security, Privacy and Trust*, pages 464–480. Springer.

[Collins, 2003] Collins, D. (2003). Pretesting survey instruments: An overview of cognitive methods. *Quality of Life Research*, 12(3):229–238.

[Connolly and Lang, 2012] Connolly, L. and Lang, M. (2012). Investigation of Cultural Aspects within Information Systems Security Research. In *ICITST '12: Proceedings of the 7th International Conference for Internet Technology and Secured Transactions*, pages 105–111. IEEE Computer Society.

[Conradi and Fuggetta, 2002] Conradi, R. and Fuggetta, A. (2002). Improving Software Process Improvement. *IEEE Software*, 19(4):92–99.

[Conradi and Wang, 2003] Conradi, R. and Wang, A. I. (2003). Introduction. In Conradi, R. and Wang, A. I., editors, *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, volume 2765, pages 1–6. Springer.

[Cooper and Schindler, 2014] Cooper, D. R. and Schindler, P. S. (2014). *Business Research Methods*. McGraw-Hill, twelfth edition.

[Cooper et al., 2009] Cooper, S., Nickell, C., Piotrowski, V., Oldfield, B., Abdallah, A., Bishop, M., Caelli, B., Dark, M., Hawthorne, E. K., Hoffman, L., Pérez, L. C., Pfleeger, C., Raines, R., Schou, C., and Brynielsson, J. (2009). An Exploration of the Current State of Information Assurance Education. *ACM SIGCSE Bulletin*, 41(4):109–125.

[Copeland, 2003] Copeland, L. (2003). *A Practitioner's Guide to Software Test Design*. Artech House.

[Coram and Bohner, 2005] Coram, M. and Bohner, S. (2005). The Impact of Agile Methods on Software Project Management. In *ECBS '05: Proceedings of 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 363–370. IEEE Computer Society.

[Crawford and Leonard, 2012] Crawford, J. and Leonard, L. N. K. (2012). Predicting post-meeting work activity in software development projects. *Team Performance Management: An International Journal*, 18(1/2):59–77.

[Cressman, 2009] Cressman, D. (2009). A Brief Overview of Actor-Network Theory: Punctualization, Heterogeneous Engineering & Translation. `http://summit.sfu.ca/system/files/iritems1/13593/0901.pdf`. Retrieved: 16th of January, 2020.

[Cressman, 2012] Cressman, D. M. (2012). *The Concert Hall as a Medium of Musical Culture: The Technical Mediation of Listening in the 19th Century*. PhD thesis, Simon Fraser University.

[Creswell and Creswell, 2018] Creswell, J. W. and Creswell, J. D. (2018). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, fifth edition.

[Creswell and Plano Clark, 2011] Creswell, J. W. and Plano Clark, V. L. (2011). *Designing and Conducting Mixed Methods Research*. SAGE Publications, second edition.

[Criscuolo et al., 2014] Criscuolo, C., Gal, P. N., and Menon, C. (2014). The Dynamics of Employment Growth: New Evidence from 18 Countries. `https://doi.org/10.1787/5jz417hj6hg6-en`. OECD Science, Technology and Industry Policy Papers, No. 14, OECD Publishing. Retrieved: 16th of January, 2020.

[Crispin and Gregory, 2009] Crispin, L. and Gregory, J. (2009). *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley.

[Crowder and Friess, 2015] Crowder, J. A. and Friess, S. (2015). *Agile Project Management: Managing for Success*. Springer.

[Crowston, 2015] Crowston, K. (2015). Rejoinder to Open Access: The Whipping Boy for Problems in Scholarly Publishing. *Communications of the Association for Information Systems*, 34:357–365.

[Cruz-Cunha, 2010] Cruz-Cunha, M. M., editor (2010). *Enterprise Information Systems for Business Integration in SMEs: Technological, Organizational, and Social Dimensions*. IGI Global.

[Cruzes et al., 2017] Cruzes, D. S., Felderer, M., Oyetoyan, T. D., Gander, M., and Pekaric, I. (2017). How is Security Testing Done in Agile Teams? A Cross-Case Analysis of Four Software Teams. In Baumeister, H., Lichter, H., and Riebisch, M., editors, *XP '17: Proceedings of the 18th International Conference on Agile Processes in Software Engineering and Extreme Programming*, pages 201–216. Springer.

[Cummings and Patel, 2009] Cummings, L. and Patel, C. (2009). *Managerial Attitudes Toward a Stakeholder Prominence within a Southeast Asia Context*. Emerald Group Publishing Limited.

[Cusumano et al., 2003] Cusumano, M., MacCormack, A., Kemerer, C. F., and Crandall, B. (2003). Software Development Worldwide: The State of the Practice. *IEEE Software*, 20(6):28–34.

[da Mota Silveira Neto et al., 2011] da Mota Silveira Neto, P. A., do Carmo Machado, I., McGregor, J. D., de Almeida, E. S., and de Lemos Meira, S. R. (2011). A systematic mapping study of software product lines testing. *Information and Software Technology*, 53(5):407–423.

[Da Veiga and Eloff, 2010] Da Veiga, A. and Eloff, J. H. P. (2010). A framework and assessment instrument for information security culture. *Computers & Security*, 29(2):196–207.

[Dalal, 2003] Dalal, S. R. (2003). Software Reliability Models: A Selective Survey and New Directions. In Pham, H., editor, *Handbook of Reliability Engineering*, chapter 11, pages 201–211. Springer.

[Daly et al., 1995] Daly, J., Miller, J., Brooks, A., Roper, M., and Wood, M. (1995). Issues on the Object-Oriented Paradigm: A Questionnaire Survey. Technical report, University of Strathclyde RR/95/183 [EFoCS-8-95].

[Dasso and Funes, 2007] Dasso, A. and Funes, A., editors (2007). *Verification, Validation and Testing in Software Engineering*. IGI Global.

[de Vries et al., 2009] de Vries, G., Visser, B., Wilhelmus, L., van Ewijk, A., van Oosterwijk, M., and Linker, B. (2009). *TPI Next: Business Driven Test Process Improvement*. UTN Publishers.

[De Win et al., 2009] De Win, B., Scandariato, R., Buyens, K., Grégoire, J., and Joosen, W. (2009). On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology*, 51(7):1152–1171.

[Deak and Stålhane, 2013] Deak, A. and Stålhane, T. (2013). Organization of Testing Activities in Norwegian Software Companies. In *ICSTW '13: Proceedings of the 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, pages 102–107. IEEE Computer Society.

[Deeg and Schreiber, 2009] Deeg, M. and Schreiber, S. (2009). Cryptographically Secure? SySS Cracks a USB Flash Drive. `https://www.syss.de/fileadmin/dokumente/Publikationen/2009/SySS_Cracks_SanDisk_USB_Flash_Drive.pdf`. Retrieved: 16th of January, 2020.

[DeFranco and Laplante, 2017] DeFranco, J. F. and Laplante, P. A. (2017). A content analysis process for qualitative software engineering research. *Innovations in Systems and Software Engineering*, 13(2):129–141.

[DeMaio et al., 1998] DeMaio, T. J., Rothgeb, J., and Hess, J. (1998). Improving Survey Quality Through Pretesting. In *Proceedings of the Survey Research Methods Section*, pages 50–58. American Statistical Association.

[DeMarco and Lister, 1999] DeMarco, T. and Lister, T. (1999). *Peopleware: Productive Projects and Teams*. Dorset House Publishing, second edition.

[Denning, 2018] Denning, S. (2018). *The Age of Agile: How Smart Companies Are Transforming the Way Work Gets Done*. American Management Association.

[Dennis, Jr., 2003] Dennis, Jr., W. J. (2003). Raising Response Rates in Mail Surveys of Small Business Owners: Results of an Experiment. *Journal of Small Business Management*, 41(3):278–295.

[Denscombe, 2010] Denscombe, M. (2010). *The Good Research Guide: For small-scale social research projects*. Open University Press, fourth edition.

[Department for Business, Energy & Industrial Strategy, 2017] Department for Business, Energy & Industrial Strategy (2017). Business population estimate for the UK and regions: 2017 statistical release. `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/663235/bpe_2017_statistical_release.pdf`. Retrieved: 16th of January, 2020.

[Department for Business, Innovation & Skills, 2011] Department for Business, Innovation & Skills (2011). Software and IT Services. https://webarchive.nationalarchives.gov.uk/20110907173723/http://www.bis.gov.uk/policies/business-sectors/electronics-and-it-services/software-and-it-services. Retrieved: 16th of January, 2020.

[Department for Business, Innovation & Skills, 2013] Department for Business, Innovation & Skills (2013). 2013 Information Security Breaches Survey: Technical Report. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/200455/bis-13-p184-2013-information-security-breaches-survey-technical-report.pdf. Retrieved: 16th of January, 2020.

[Desikan and Ramesh, 2006] Desikan, S. and Ramesh, G. (2006). *Software Testing: Principles and Practices*. Pearson Education.

[Devito Da Cunha and Greathead, 2007] Devito Da Cunha, A. and Greathead, D. (2007). Does personality matter? An analysis of code-review ability. *Communications of the ACM*, 50(5):109–112.

[Dey et al., 2010] Dey, D., Fan, M., and Zhang, C. (2010). Design and Analysis of Contracts for Software Outsourcing. *Information Systems Research*, 21(1):93–114.

[Dhaliwal et al., 2011] Dhaliwal, J., Onita, C. G., Poston, R., and Zhang, X. P. (2011). Alignment within the software development unit: Assessing structural and relational dimensions between developers and testers. *The Journal of Strategic Information Systems*, 20(4):323–342.

[Di Penta and Tamburri, 2017] Di Penta, M. and Tamburri, D. A. (2017). Combining Quantitative and Qualitative Studies in Empirical Software Engineering Research. In *ICSE-C '17: Proceedings of the 39th International Conference on Software Engineering Companion*, pages 499–500. IEEE Computer Society.

[Dias-Neto et al., 2017] Dias-Neto, A. C., Matalonga, S., Solari, M., Robiolo, G., and Travassos, G. H. (2017). Toward the characterization of software testing practices in South America: looking at Brazil and Uruguay. *Software Quality Journal*, 25(4):1145–1183.

[Diebold et al., 2015] Diebold, P., Ostberg, J.-P., Wagner, S., and Zendler, U. (2015). What Do Practitioners Vary in Using Scrum? In Lassenius, C., Dingsøyr, T., and Paasivaara, M., editors, *XP '15: Proceedings of the 16th International Conference*

on Agile Processes in Software Engineering and Extreme Programming, pages 40–51. Springer.

[Dietl et al., 2012] Dietl, W., Dietzel, S., Ernst, M. D., Mote, N., Walker, B., Cooper, S., Pavlik, T., and Popović, Z. (2012). Verification Games: Making Verification Fun. In *FTfJP '12: Proceedings of the 14th Workshop on Formal Techniques for Java-like Programs*, pages 42–49. ACM Press.

[Dimopoulos et al., 2004] Dimopoulos, V., Furnell, S., Jennex, M., and Kritharas, I. (2004). Approaches to IT Security in Small and Medium Enterprises. In *Proceedings of the 2nd Australian Information Security Management Conference*, pages 73–82. School of Computer & Information Science, Edith Cowan University.

[do Carmo Machado et al., 2014] do Carmo Machado, I., McGregor, J. D., Cavalcanti, Y. C., and de Almeida, E. S. (2014). On strategies for testing software product lines: A systematic literature review. *Information and Software Technology*, 56(10):1183–1199.

[Dojkovski et al., 2006] Dojkovski, S., Lichtenstein, S., and Warren, M. (2006). Challenges in Fostering an Information Security Culture in Australian Small and Medium Sized Enterprises. In *ECIW '06: Proceedings of the 5th European Conference on Information Warfare and Security*, pages 31–40. Academic Conferences Limited.

[Dojkovski et al., 2007] Dojkovski, S., Lichtenstein, S., and Warren, M. (2007). Fostering Information Security Culture in Small and Medium Size Enterprises: An Interpretive Study in Australia. In *ECIS '07: Proceedings of the 15th European Conference on Information Systems*, pages 1560–1571. University of St. Gallen.

[Dojkovski et al., 2010] Dojkovski, S., Lichtenstein, S., and Warren, M. J. (2010). Enabling Information Security Culture: Influences and Challenges for Australian SMEs. In *ACIS '10: Proceedings of the 21st Australasian Conference on Information Systems*. AIS Electronic Library.

[Doolin and McLeod, 2012] Doolin, B. and McLeod, L. (2012). Sociomateriality and boundary objects in information systems development. *European Journal of Information Systems*, 21(5):570–586.

[Doyle et al., 2016] Doyle, L., Brady, A.-M., and Byrne, G. (2016). An overview of mixed methods research - revisited. *Journal of Research in Nursing*, 21(8):623–635.

[Drennan, 2003] Drennan, J. (2003). Cognitive interviewing: verbal data in the design and pretesting of questionnaires. *Journal of Advanced Nursing*, 42(1):57–63.

[Duan et al., 2002] Duan, Y., Mullins, R., Hamblin, D., Stanek, S., Sroka, H., Machado, V., and Araujo, J. (2002). Addressing ICTs skill challenges in SMEs: insights from three country investigations. *Journal of European Industrial Training*, 26(9):430–441.

[Duncan and Whittington, 2014] Duncan, B. and Whittington, M. (2014). Compliance with standards, assurance and audit: does this equal security? In *SIN '14: Proceedings of the 7th International Conference on Security of Information and Networks*. ACM Press.

[Dunsmore et al., 2001] Dunsmore, A., Roper, M., and Wood, M. (2001). Systematic Object-Oriented Inspection - An Empirical Study. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 135–144. IEEE Computer Society.

[Dustin, 2002] Dustin, E. (2002). *Effective Software Testing: 50 Specific Ways to Improve Your Testing*. Addison-Wesley.

[Dustin et al., 1999] Dustin, E., Rashka, J., and Paul, J. (1999). *Automated Software Testing: Introduction, Management, and Performance*. Addison-Wesley.

[Dybå, 2003] Dybå, T. (2003). Factors of Software Process Improvement Success in Small and Large Organizations: An Empirical Study in the Scandinavian Context. *ACM SIGSOFT Software Engineering Notes*, 28(5):148–157.

[Dybå et al., 2007] Dybå, T., Dingsøyr, T., and Hanssen, G. K. (2007). Applying Systematic Reviews to Diverse Study Types: An Experience Report. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 225–234. IEEE Computer Society.

[Dybå et al., 2004] Dybå, T., Dingsøyr, T., and Moe, N. B. (2004). *Process Improvement in Practice: A Handbook for IT Companies*. Kluwer Academic Publishers.

[Dybkaer, 2011] Dybkaer, R. (2011). 'Verification' versus 'Validation': A Terminological Comparison. *Accreditation and Quality Assurance*, 16(2):105–108.

[Dyerson et al., 2016] Dyerson, R., Spinelli, R., and Harindranath, G. (2016). Revisiting IT readiness: an approach for small firms. *Industrial Management & Data Systems*, 116(3):546–563.

[Eason, 2008] Eason, K. (2008). Sociotechnical systems theory in the 21st Century: another half-filled glass? In Graves, D., editor, *Sense in Social Science: A collection of essays in honour of Dr. Lisl Klein*, pages 123–134. Desmond Graves.

[Easterbrook et al., 2008] Easterbrook, S., Singer, J., Storey, M.-A., and Damian, D. (2008). Selecting Empirical Methods for Software Engineering Research. In Shull, F., Singer, J., and Sjøberg, D. I. K., editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer.

[Edmundson et al., 2013] Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A., and Wagner, D. (2013). An Empirical Study on the Effectiveness of Security Code Review. In Jürjens, J., Livshits, B., and Scandariato, R., editors, *Engineering Secure Software and Systems*, volume 7781 of *Lecture Notes in Computer Science*, pages 197–212. Springer.

[Eeles and Cripps, 2010] Eeles, P. and Cripps, P. (2010). *The Process of Software Architecting*. Addison-Wesley.

[Egorova et al., 2009] Egorova, E., Torchiano, M., and Morisio, M. (2009). Evaluating the Perceived Effect of Software Engineering Practices in the Italian Industry. In *ICSP '09: Proceedings of the International Conference on Software Process: Trustworthy Software Development Processes*, pages 100–111. Springer.

[Elbanna, 2016] Elbanna, A. (2016). Doing Sociomateriality Research in Information Systems. *The Data Base for Advances in Information Systems*, 47(4):84–92.

[Elbanna, 2018] Elbanna, A. (2018). Making a Difference in ICT Research: Feminist Theorization of Sociomateriality and the Diffraction Methodology. In Schultze, U., Aanestad, M., Mähring, M., Østerlund, C., and Riemer, K., editors, *Living with Monsters? Social Implications of Algorithmic Phenomena, Hybrid Agency, and the Performativity of Technology*, volume 543 of *IFIP Advances in Information and Communication Technology*, pages 140–155. Springer.

[Elbaum et al., 2007] Elbaum, S., Person, S., Dokulil, J., and Jorde, M. (2007). Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable. In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*, pages 688–697. IEEE Computer Society.

[Ellims et al., 2004] Ellims, M., Bridges, J., and Ince, D. C. (2004). Unit Testing in Practice. In *ISSRE '04: Proceedings of the 15th International Symposium on Software Reliability Engineering*, pages 3–13. IEEE Computer Society.

[Engel, 2010] Engel, A. (2010). *Verification, Validation and Testing of Engineered Systems*. John Wiley & Sons.

[Engström and Runeson, 2010] Engström, E. and Runeson, P. (2010). A Qualitative Survey of Regression Testing Practices. In Babar, M. A., Vierimaa, M., and Oivo, M., editors, *Product-Focused Software Process Improvement*, volume 6156 of *Lecture Notes in Computer Science*, pages 3–16. Springer.

[Epstein, 2009] Epstein, J. (2009). A Survey of Vendor Software Assurance Practices. In *ACSAC '09: Proceedings of the 25th Annual Computer Security Applications Conference*, pages 528–537. IEEE Computer Society.

[Eschelbeck, 2005] Eschelbeck, G. (2005). The Laws of Vulnerabilities: Which security vulnerabilities really matter? *Information Security Technical Report*, 10(4):213–219.

[Estimé et al., 1993] Estimé, M.-F., Drilhon, G., and Julien, P.-A. (1993). *Small and Medium-sized Enterprises: Technology and Competitiveness*. Organisation for Economic Co-operation and Development.

[European Commission, 2015] European Commission (2015). User guide to the SME definition. `https://ec.europa.eu/docsroom/documents/15582/attachments/1/translations/en/renditions/pdf`. Retrieved: 16th of January, 2020.

[Evans and Mathur, 2005] Evans, J. R. and Mathur, A. (2005). The value of online surveys. *Internet Research*, 15(2):195–219.

[Everett and McLeod, Jr., 2007] Everett, G. D. and McLeod, Jr., R. (2007). *Software Testing: Testing Across the Entire Software Development Life Cycle*. John Wiley & Sons.

[Experimentus Ltd, 2012] Experimentus Ltd (2012). Test Maturity Model integrated (TMMi) Survey results: How Mature Are Companies' Software Quality Management Processes In Today's Market? Technical report, Experimentus Ltd.

[Experimentus Ltd, 2020] Experimentus Ltd (2020). TMMi Industry Survey. `http://experimentus.tmmibenchmark.sgizmo.com/s3/`. Retrieved: 16th of January, 2020.

[Fagan, 1976] Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211.

[Fagan, 1986] Fagan, M. E. (1986). Advances in Software Inspections. *IEEE Transactions on Software Engineering*, 12(7):744–751.

[Fairley, 2009] Fairley, R. E. (2009). *Managing and Leading Software Projects*. John Wiley & Sons.

[Fan and Yan, 2010] Fan, W. and Yan, Z. (2010). Factors affecting response rates of the web survey: A systematic review. *Computers in Human Behavior*, 26(2):132–139.

[Favre et al., 2003] Favre, J.-M., Estublier, J., and Sanlaville, R. (2003). Tool Adoption Issues in a Very Large Software Company. In *ACSE '03: Proceedings of the 3rd International Workshop on Adoption Centric Software Engineering*, pages 81–89.

[Fayad et al., 2000] Fayad, M. E., Laitinen, M., and Ward, R. P. (2000). Software Engineering in the Small. *Communications of the ACM*, 43(3):115–118.

[FDA, 2002] FDA (2002). General Principles of Software Validation; Final Guidance for Industry and FDA Staff. `https://www.fda.gov/media/73141/download`. Retrieved: 16th of January, 2020.

[Felderer et al., 2011] Felderer, M., Agreiter, B., Zech, P., and Breu, R. (2011). A Classification for Model-Based Security Testing. In *VALID '11: Proceedings of the 3rd International Conference on Advances in System Testing and Validation Lifecycle*, pages 109–114. Xpert Publishing Services.

[Feldmann and Pizka, 2003] Feldmann, R. L. and Pizka, M. (2003). An On-Line Software Engineering Repository for Germany's SME - An Experience Report. In Henninger, S. and Maurer, F., editors, *Advances in Learning Software Organizations*, volume 2640 of *Lecture Notes in Computer Science*, pages 34–43. Springer.

[Fernández-Sanz et al., 2009] Fernández-Sanz, L., Villalba, M. T., Hilera, J. R., and Lacuesta, R. (2009). Factors with Negative Influence on Software Testing Practice in Spain: A Survey. In O'Connor, R. V., Baddoo, N., Gallego, J. C., Muslera, R. R., Smolander, K., and Messnarz, R., editors, *Software Process Improvement*, volume 42, pages 1–12. Springer.

[Fewster and Graham, 1999] Fewster, M. and Graham, D. (1999). *Software Test Automation: Effective Use of Test Execution Tools*. ACM Press/Addison-Wesley.

[Fink, 1994] Fink, D. (1994). A Security Framework for Information Systems Outsourcing. *Information Management & Computer Security*, 2(4):3–8.

[Fischer-Hübner, 2001] Fischer-Hübner, S. (2001). *IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms*. Springer.

[Fitzgerald and Howcroft, 1998] Fitzgerald, B. and Howcroft, D. (1998). Competing Dichotomies in IS Research and Possible Strategies for Resolution. In *ICIS '98: Proceedings of the 19th International Conference on Information Systems*, pages 155–164.

[Flavián and Guinalíu, 2006] Flavián, C. and Guinalíu, M. (2006). Consumer Trust, Perceived Security and Privacy Policy: Three Basic Elements of Loyalty to a Web Site. *Industrial Management & Data Systems*, 106(5):601–620.

[Foote, 2004] Foote, D. (2004). Recipe for offshore outsourcing failure: Ignore organization, people issues. *ABA Banking Journal*, 96(9):56–59.

[Forsgren and Kersten, 2018] Forsgren, N. and Kersten, M. (2018). DevOps Metrics. *Communications of the ACM*, 61(4):44–48.

[Fowler, Jr., 1995] Fowler, Jr., F. J. (1995). *Improving Survey Questions: Design and Evaluation*. SAGE Publications.

[Fowler, Jr., 2009] Fowler, Jr., F. J. (2009). *Survey Research Methods*. SAGE Publications, fourth edition.

[Fox, 2006] Fox, C. (2006). *Introduction to Software Engineering Design: Processes, Principles, and Patterns with UML2*. Addison-Wesley.

[França et al., 2014] França, C., Sharp, H., and da Silva, F. Q. B. (2014). Motivated software engineers are engaged and focused, while satisfied ones are happy. In *ESEM '14: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM Press.

[Frederiksen and Rose, 2003] Frederiksen, H. D. and Rose, J. (2003). The social construction of the software operation: Reinforcing effects in metrics programs. *Scandinavian Journal of Information Systems*, 15(1):23–37.

[Freeman, 2010] Freeman, R. E. (2010). *Strategic Management: A Stakeholder Approach*. Cambridge University Press.

[Furnell, 2007] Furnell, S. (2007). IFIP workshop - Information security culture. *Computers & Security*, 26(1):35.

[Futcher and von Solms, 2008] Futcher, L. and von Solms, R. (2008). Guidelines for Secure Software Development. In *SAICSIT '08: Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 56–65. ACM Press.

[Futrell et al., 2002] Futrell, R. T., Shafer, D. F., and Shafer, L. I. (2002). *Quality Software Project Management*. Prentice Hall.

[Gad and Jensen, 2010] Gad, C. and Jensen, C. B. (2010). On the Consequences of Post-ANT. *Science, Technology, & Human Values*, 35(1):55–80.

[Galen, 2005] Galen, R. (2005). *Software Endgames: Eliminating Defects, Controlling Change, and the Countdown to On-time Delivery*. Dorset House Publishing.

[Gao et al., 2003] Gao, J. Z., Tsao, H.-S. J., and Wu, Y. (2003). *Testing and Quality Assurance for Component-Based Software*. Artech House.

[Garousi and Felderer, 2017] Garousi, V. and Felderer, M. (2017). Worlds Apart: Industrial and Academic Focus Areas in Software Testing. *IEEE Software*, 34(5):38–45.

[Garousi and Varma, 2010] Garousi, V. and Varma, T. (2010). A Replicated Survey of Software Testing Practices in the Canadian Province of Alberta: What has Changed from 2004 to 2009? Technical report, University of Calgary.

[Geer, Jr. et al., 2003] Geer, Jr., D., Hoo, K. S., and Jaquith, A. (2003). Information Security: Why the Future Belongs to the Quants. *IEEE Security & Privacy*, 1(4):24–32.

[Geras et al., 2004] Geras, A. M., Smith, M. R., and Miller, J. (2004). A Survey of Software Testing Practices in Alberta. *Canadian Journal of Electrical and Computer Engineering*, 29(3):183–191.

[Ghag, 2008] Ghag, A. (2008). Case Study: Testing for the Utilities Sector. In Hendel, A., Messner, W., and Thun, F., editors, *Rightshore!: Successfully Industrialize SAP® Projects Offshore*, chapter 12, pages 187–201. Springer.

[Gibbs, 2018] Gibbs, G. R. (2018). *Analyzing Qualitative Data*. SAGE Publications, second edition.

[Giddens, 1984] Giddens, A. (1984). *The Constitution of Society: Outline of the Theory of Structuration*. Polity Press.

[Giddens, 1991] Giddens, A. (1991). Structuration theory: past, present and future. In Bryant, C. G. A. and Jary, D., editors, *Giddens' Theory of Structuration: A critical appreciation*, chapter 8, pages 201–221. Routledge.

[Gilbert, 2011] Gilbert, D. (2011). A Nimble Test Plan: Removing the Cost of Overplanning. In Heusser, M. and Kulkarni, G., editors, *How to Reduce the Cost of Software Testing*, chapter 12, pages 179–194. CRC Press.

[Gillham, 2008] Gillham, B. (2008). *Developing a Questionnaire*. Continuum, second edition.

[Ginsberg and Quinn, 1995] Ginsberg, M. P. and Quinn, L. H. (1995). Process Tailoring and the the Software Capability Maturity Model. Technical report, Carnegie Mellon University CMU/SEI-94-TR-024.

[Gittens et al., 2002] Gittens, M., Lutfiyya, H., Bauer, M., Godwin, D., Kim, Y. W., and Gupta, P. (2002). An Empirical Evaluation of System and Regression Testing. In *CASCON '02: Proceedings of the 2002 Conference of the Centre for Advanced Studies on Collaborative Research*. IBM Press.

[Gleirscher et al., 2012] Gleirscher, M., Golubitskiy, D., Irlbeck, M., and Wagner, S. (2012). On the Benefit of Automated Static Analysis for Small and Medium-Sized Software Enterprises. In Biffl, S., Winkler, D., and Bergsmann, J., editors, *Software Quality. Process Automation in Software Development*, volume 94 of *Lecture Notes in Business Information Processing*, pages 14–38. Springer.

[Glogowska, 2011] Glogowska, M. (2011). Paradigms, pragmatism and possibilities: mixed-methods research in speech and language therapy. *International Journal of Language & Communication Disorders*, 46(3):251–260.

[Goertzel et al., 2007] Goertzel, K. M., Winograd, T., McKinley, H. L., Oh, L., Colon, M., McGibbon, T., Fedchak, E., and Vienneau, R. (2007). Software Security Assurance: A State-of-the-Art Report (SOAR). Technical report, Information Assurance Technology Analysis Center (IATAC) and Data and Analysis Center for Software (DACS).

[Gök et al., 2015] Gök, A., Waterworth, A., and Shapira, P. (2015). Use of web mining in studying innovation. *Scientometrics*, 102(1):653–671.

[Gokhale and Banks, 2004] Gokhale, G. B. and Banks, D. A. (2004). Organisational Information Security: A Viable System Perspective. In *Proceedings of the 2nd Australian Information Security Management Conference*, pages 178–184. School of Computer & Information Science, Edith Cowan University.

[Goldkuhl, 2012] Goldkuhl, G. (2012). Pragmatism vs interpretivism in qualitative information systems research. *European Journal of Information Systems*, 21(2):135–146.

[Gollmann, 2011] Gollmann, D. (2011). *Computer Security*. John Wiley & Sons, third edition.

[Goodman, 2005] Goodman, F. A. (2005). *Defining and Deploying Software Processes*. Auerbach Publications.

[Google, 2020] Google (2020). Google Scholar. `https://scholar.google.co.uk/`. Retrieved: 16th of January, 2020.

[Goth, 2009] Goth, G. (2009). Agile Tool Market Growing with the Philosophy. *IEEE Software*, 26(2):88–91.

[Grant, 2013] Grant, K. A. (2013). Outsourcing Security: Alliance Portfolio Size, Capability, and Reliability. *International Studies Quarterly*, 57(2):418–429.

[Greanier, 2004] Greanier, T. (2004). *Java Foundations*. John Wiley & Sons.

[Greene et al., 1989] Greene, J. C., Caracelli, V. J., and Graham, W. F. (1989). Toward a Conceptual Framework for Mixed-Method Evaluation Designs. *Educational Evaluation and Policy Analysis*, 11(3):255–274.

[Greener, 2008] Greener, I. (2008). Expert Patients and Human Agency: Long-term Conditions and Giddens' Structuration Theory. *Social Theory & Health*, 6(4):273–290.

[Gregory, 2015] Gregory, P. H. (2015). *CISSP Guide to Security Essentials*. Cengage Learning, second edition.

[Gren and Antinyan, 2017] Gren, L. and Antinyan, V. (2017). On the Relation Between Unit Testing and Code Quality. In *SEAA '17: Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications*, pages 52–56. IEEE Computer Society.

[Griffith and Dougherty, 2002] Griffith, T. L. and Dougherty, D. J. (2002). Beyond socio-technical systems: introduction to the special issue. *Journal of Engineering and Technology Management*, 19(2):205–216.

[Grindal et al., 2006a] Grindal, M., Offutt, J., and Mellin, J. (2006a). On the Testing Maturity of Software Producing Organizations. In *TAIC-PART '06: Proceedings of the Testing: Academic & Industrial Conference on Practice And Research Techniques*, pages 171–180. IEEE Computer Society.

[Grindal et al., 2006b] Grindal, M., Offutt, J., and Mellin, J. (2006b). On the Testing Maturity of Software Producing Organizations: Detailed Data. Technical report, George Mason University ISE-TR-06-03.

[Groves et al., 2000] Groves, L., Nickson, R., Reeve, G., Reeves, S., and Utting, M. (2000). A Survey of Software Development Practices in the New Zealand Software Industry. In *ASWEC '00: Proceedings of the 2000 Australian Software Engineering Conference*, pages 189–202. IEEE Computer Society.

[Guest et al., 2006] Guest, G., Bunce, A., and Johnson, L. (2006). How Many Interviews Are Enough?: An Experiment with Data Saturation and Variability. *Field Methods*, 18(1):59–82.

[Habra et al., 1999] Habra, N., Niyitugabira, E., Lamblin, A.-C., and Renault, A. (1999). Software Process Improvement in Small Organizations Using Gradual Evaluation Schema. In *PROFES '99: Proceedings of the 1st International Conference on Product Focused Software Process Improvement*, pages 381–396.

[Halaweh and Fidler, 2008] Halaweh, M. and Fidler, C. (2008). Security Perception in E-commerce: Conflict between Customer and Organizational Perspectives. In *IMCSIT '08: Proceedings of the International Multiconference on Computer Science and Information Technology*, pages 443–449. IEEE Computer Society.

[Hall et al., 2001] Hall, T., Rainer, A., Baddoo, N., and Beecham, S. (2001). An Empirical Study of Maintenance Issues within Process Improvement Programmes in the Software Industry. In *ICSM '01: Proceedings of the 2001 IEEE International Conference on Software Maintenance*, pages 422–430. IEEE Computer Society.

[Hall et al., 2008] Hall, T., Sharp, H., Beecham, S., Baddoo, N., and Robinson, H. (2008). What Do We Know about Developer Motivation? *IEEE Software*, 25(4):92–94.

[Hamlet, 2010] Hamlet, D. (2010). *Composing Software Components: A Software-testing Perspective*. Springer.

[Hamlet, 1988] Hamlet, R. (1988). Special Section on Software Testing. *Communications of the ACM*, 31(6):662–667.

[Han et al., 2009] Han, J., Gao, D., and Deng, R. H. (2009). On the Effectiveness of Software Diversity: A Systematic Study on Real-World Vulnerabilities. In Flegel, U. and Bruschi, D., editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 5587 of *Lecture Notes in Computer Science*, pages 127–146. Springer.

[Harrer and Wald, 2016] Harrer, J. and Wald, A. (2016). Levers of enterprise security control: a study on the use, measurement and value contribution. *Journal of Management Control*, 27(1):7–32.

[Harrington, 2006] Harrington, H. J. (2006). *Process Management Excellence: The Art of Excelling in Process Management*. Paton Press.

[Harrison et al., 1999] Harrison, R., Badoo, N., Barry, E., Biffl, S., Parra, A., Winter, B., and Wuest, J. (1999). Directions and Methodologies for Empirical Software Engineering Research. *Empirical Software Engineering*, 4(4):405–410.

[Harrold, 2000] Harrold, M. J. (2000). Testing: A Roadmap. In *ICSE '00: Proceedings of the 22nd International Conference on Software Engineering - Future of Software Engineering Track*, pages 61–72. ACM Press.

[Hartman, 2002] Hartman, A. (2002). Is ISSTA Research Relevant to Industry? *ACM SIGSOFT Software Engineering Notes*, 27:205–206.

[Harvie and Lee, 2002] Harvie, C. and Lee, B.-C. (2002). East Asian SMEs: Contemporary Issues and Developments - An Overview. In Harvie, C. and Lee, B.-C., editors, *The Role of SMEs in National Economies in East Asia*, chapter 1, pages 1–20. Edward Elgar Publishing Ltd.

[Hashmi et al., 2010] Hashmi, Z., Shaikh, S. A., and Ikram, N. (2010). Methodologies and Tools for OSS Software: Current State of the Practice. In *OpenCert '10: Proceedings of the 4th International Workshop on Foundations and Techniques for Open Source Software Certification*, pages 104–115.

[Hass, 2014] Hass, A. M. (2014). *Guide to Advanced Software Testing*. Artech House, second edition.

[Hass, 2008] Hass, A. M. J. (2008). *Guide to Advanced Software Testing*. Artech House.

[Hauck et al., 2008] Hauck, J. C. R., Gresse von Wangenheim, C., de Souza, R. H., and Thiry, M. (2008). Process Reference Guides - Support for Improving Software Processes in Alignment with Reference Models and Standards. In O'Connor, R. V., Baddoo, N., Smolander, K., and Messnarz, R., editors, *Software Process Improvement*, volume 16, pages 70–81. Springer.

[Hax, 2010] Hax, A. C. (2010). Managing Small- and Medium-Sized Enterprises (SMEs) - Lessons from the Delta Model. In *The Delta Model: Reinventing Your Business Strategy*, pages 167–181. Springer New York.

[Hazzan and Dubinsky, 2008] Hazzan, O. and Dubinsky, Y. (2008). *Agile Software Engineering*. Springer.

[Hazzan and Tomayko, 2004] Hazzan, O. and Tomayko, J. (2004). Human Aspects of Software Engineering: The Case of Extreme Programming. In Eckstein, J. and Baumeister, H., editors, *XP '04: Proceedings of the 5th International Conference on Extreme Programming and Agile Processes in Software Engineering*, pages 303–311. Springer.

[Heerwegh et al., 2005] Heerwegh, D., Vanhove, T., Matthijs, K., and Loosveldt, G. (2005). The Effect of Personalization on Response Rates and Data Quality in Web Surveys. *International Journal of Social Research Methodology*, 8(2):85–99.

[Hendry et al., 2000] Hendry, C., Brown, J., and Defillippi, R. (2000). Understanding Relationships between Universities and SMEs in Emerging High Technology Industries: The Case of Opto-Electronics. *International Journal of Innovation Management*, 4(1):51–75.

[Henry, 2005] Henry, T. R. (2005). Software Development Productivity: Considering the Socio-Technical Side of Software Development. *Issues in Information Systems*, 6(2):110–116.

[Hetzel, 1976] Hetzel, W. C. (1976). *An Experimental Analysis of Program Verification Methods*. PhD thesis, The University of North Carolina at Chapel Hill.

[Hoda et al., 2008] Hoda, R., Noble, J., and Marshall, S. (2008). A for Agile, Issues with Awareness and Adoption. In *AGILE '08: Proceedings of the Research-In-Progress Track at the AGILE 2008 Conference*.

[Höfer and Tichy, 2007] Höfer, A. and Tichy, W. F. (2007). Status of Empirical Research in Software Engineering. In Basili, V. R., Rombach, D., Schneider, K., Kitchenham, B., Pfahl, D., and Selby, R. W., editors, *Empirical Software Engineering Issues: Critical Assessment and Future Directions*, volume 4336, pages 10–19. Springer.

[Holeman, 2018] Holeman, I. (2018). Room for Silence: Ebola Research, Pluralism and the Pragmatic Study of Sociomaterial Practices. *Computer Supported Cooperative Work*, 27(3):389–425.

[Horton et al., 2005] Horton, K., Davenport, E., and Wood-Harper, T. (2005). Exploring sociotechnical interaction with Rob Kling: five "big" ideas. *Information Technology & People*, 18(1):50–67.

[Hove and Anda, 2005] Hove, S. E. and Anda, B. (2005). Experiences from Conducting Semi-Structured Interviews in Empirical Software Engineering Research. In *METRICS '05: Proceedings of the 11th International Software Metrics Symposium*. IEEE Computer Society.

[Howard and Lipner, 2006] Howard, M. and Lipner, S. (2006). *The Security Development Lifecycle*. Microsoft Press.

[Howcroft and Light, 2010] Howcroft, D. and Light, B. (2010). The Social Shaping of Packaged Software Selection. *Journal of the Association for Information Systems*, 11(3):122–148.

[Howcroft et al., 2004] Howcroft, D., Mitev, N., and Wilson, M. (2004). What We May Learn from the Social Shaping of Technology Approach. In Mingers, J. and Willcocks, L., editors, *Social Theory and Philosophy for Information Systems*, chapter 9, pages 329–371. John Wiley & Sons.

[Howe, 1988] Howe, K. R. (1988). Against the Quantitative-Qualitative Incompatibility Thesis or Dogmas Die Hard. *Educational Researcher*, 17(8):10–16.

[Hu et al., 2012] Hu, Q., Dinev, T., Hart, P., and Cooke, D. (2012). Managing Employee Compliance with Information Security Policies: The Critical Role of Top Management and Organizational Culture. *Decision Sciences*, 43(4):615–660.

[Humayun et al., 2010] Humayun, A., Basit, W., Farrukh, G. A., Lodhi, F., and Aden, R. (2010). An Empirical Analysis of Team Review Approaches for Teaching Quality Software Development. In *ICSE '10: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, pages 567–575. ACM Press.

[Hunt et al., 2007] Hunt, A., Thomas, D., and Hargett, M. (2007). *Pragmatic Unit Testing in C# with NUnit*. Pragmatic Bookshelf, second edition.

[Hunt et al., 1982] Hunt, S. D., Sparkman, Jr., R. D., and Wilcox, J. B. (1982). The Pretest in Survey Research: Issues and Preliminary Findings. *Journal of Marketing Research*, 19(2):269–273.

[Hutcheson, 2003] Hutcheson, M. L. (2003). *Software Testing Fundamentals: Methods and Metrics*. John Wiley & Sons.

[IEEE, 1990] IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990.

[IEEE, 2008] IEEE (2008). IEEE Standard for Software and System Test Documentation. IEEE Std 829-2008.

[IEEE, 2020] IEEE (2020). IEEE Xplore Digital Library. `https://ieeexplore.ieee.org/Xplore/home.jsp`. Retrieved: 16th of January, 2020.

[Ikerionwu et al., 2017] Ikerionwu, C., Edgar, D., and Gray, E. (2017). The development of service provider's BPO-IT framework. *Business Process Management Journal*, 23(5):897–917.

[Ilieva et al., 2002] Ilieva, J., Baron, S., and Healey, N. M. (2002). Online surveys in marketing research: pros and cons. *International Journal of Market Research*, 44(3):361–376.

[Illes et al., 2005] Illes, T., Herrmann, A., Paech, B., and Rückert, J. (2005). Criteria for Software Testing Tool Evaluation - A Task Oriented View. In *3WCSQ: Proceedings of the 3rd World Congress for Software Quality*. International Software Quality Institute.

[International Finance Corporation - World Bank Group, 2010] International Finance Corporation - World Bank Group (2010). The SME Banking Knowledge Guide. `https://www.ifc.org/wps/wcm/connect/c6298e7b-9a16-4925-b6c0-81ea8d2ada28/SMEE.pdf?MOD=AJPERES`. Retrieved: 16th of January, 2020.

[Itkonen et al., 2009] Itkonen, J., Mäntylä, M. V., and Lassenius, C. (2009). How Do Testers Do It? An Exploratory Study on Manual Testing Practices. In *ESEM '09: Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 494–497. IEEE Computer Society.

[Itkonen et al., 2016] Itkonen, J., Mäntylä, M. V., and Lassenius, C. (2016). Test Better by Exploring: Harnessing Human Skills and Knowledge. *IEEE Software*, 33(4):90–96.

[Ivankova et al., 2006] Ivankova, N. V., Creswell, J. W., and Stick, S. L. (2006). Using Mixed-Methods Sequential Explanatory Design: From Theory to Practice. *Field Methods*, 18(1):3–20.

[Ivarsson and Gorschek, 2011] Ivarsson, M. and Gorschek, T. (2011). A Method for Evaluating Rigor and Industrial Relevance of Technology Evaluations. *Empirical Software Engineering*, 16(3):365–395.

[Iyamu and Sehlola, 2012] Iyamu, T. and Sehlola, P. (2012). The Impact of Risk on Information Technology Projects. In *ICMIT '12: Proceedings of the 2012 International Conference on Management of Innovation and Technology*, pages 678–683. IEEE Computer Society.

[Jaatun et al., 2011] Jaatun, M. G., Jensen, J., Meland, P. H., and Tøndel, I. A. (2011). A Lightweight Approach to Secure Software Engineering. In Mjølsnes, S. F., editor, *A Multidisciplinary Introduction to Information Security*, chapter 10, pages 183–216. CRC Press.

[Jacobs and Trienekens, 2002] Jacobs, J. and Trienekens, J. (2002). Towards a Metrics Based Verification and Validation Maturity Model. In Anderson, S., Felici, M., and Bologna, S., editors, *Computer Safety, Reliability and Security*, volume 2434 of *Lecture Notes in Computer Science*, pages 175–185. Springer.

[James and Busher, 2006] James, N. and Busher, H. (2006). Credibility, authenticity and voice: dilemmas in online interviewing. *Qualitative Research*, 6(3):403–420.

[Janczewski, 2000] Janczewski, L. (2000). Managing Security Functions Using Security Standards. In Janczewski, L., editor, *Internet and Intranet Security Management: Risks and Solutions: Risks and Solutions*, chapter 4, pages 81–105. IGI Global.

[Jawadekar, 2004] Jawadekar, W. S. (2004). *Software Engineering: Principles and Practice*. McGraw-Hill.

[Jeffery and Votta, 1999] Jeffery, D. R. and Votta, L. G. (1999). Empirical Software Engineering. *IEEE Transactions on Software Engineering*, 25(4):435–437.

[Jensen, 2010] Jensen, C. J. (2010). *Discovering and Modeling Open Source Software Processes*. PhD thesis, University of California, Irvine.

[Jick, 1979] Jick, T. D. (1979). Mixing Qualitative and Quantitative Methods: Triangulation in Action. *Administrative Science Quarterly*, 24(4):602–611.

[Johnson and Weller, 2001] Johnson, J. C. and Weller, S. C. (2001). Elicitation Techniques for Interviewing. In Gubrium, J. F. and Holstein, J. A., editors, *Handbook of Interview Research: Context and Method*, chapter 24, pages 491–514. SAGE Publications.

[Johnson and Goetz, 2007] Johnson, M. E. and Goetz, E. (2007). Embedding Information Security into the Organization. *IEEE Security & Privacy*, 5(3):16–24.

[Johnson and Tjahjono, 1998] Johnson, P. M. and Tjahjono, D. (1998). Does Every Inspection Really Need a Meeting? *Empirical Software Engineering*, 3(1):9–35.

[Johnson and Onwuegbuzie, 2004] Johnson, R. B. and Onwuegbuzie, A. J. (2004). Mixed Methods Research: A Research Paradigm Whose Time Has Come. *Educational Researcher*, 33(7):14–26.

[Johri, 2011] Johri, A. (2011). Sociomaterial bricolage: The creation of location-spanning work practices by global software developers. *Information and Software Technology*, 53(9):955–968.

[Jones, 2012] Jones, C. (2012). A Short History of the Cost Per Defect Metric. `https://www.ifpug.org/content/documents/Jones-CostPerDefectMetricVersion4.pdf`. Retrieved: 16th of January, 2020.

[Jones, 2000] Jones, E. L. (2000). Software Testing in the Computer Science Curriculum – A Holistic Approach. In *ACSE '00: Proceedings of the Australasian Conference on Computing Education*, pages 153–157. ACM Press.

[Jones and Chatmon, 2001] Jones, E. L. and Chatmon, C. L. (2001). A Perspective on Teaching Software Testing. *Journal of Computing Sciences in Colleges*, 16(3):92–100.

[Jones, 2011] Jones, M. (2011). Structuration Theory. In Galliers, R. D. and Currie, W. L., editors, *The Oxford Handbook of Management Information Systems: Critical Perspectives and New Directions*, chapter 5, pages 113–136. Oxford University Press.

[Jones, 2014] Jones, M. (2014). A Matter of Life and Death: Exploring Conceptualizations of Sociomateriality in the Context of Critical Care. *MIS Quarterly*, 38(3):895–925.

[Jones and Karsten, 2008] Jones, M. R. and Karsten, H. (2008). Giddens's Structuration Theory and Information Systems Research. *MIS Quarterly*, 32(1):127–157.

[Jourdan, 2009] Jourdan, G.-V. (2009). Securing Large Applications Against Command Injections. *IEEE Aerospace and Electronics Systems Magazine*, 24(6):15–24.

[Jovanović et al., 2017] Jovanović, M., Mas, A., Mesquida, A.-L., and Lalić, B. (2017). Transition of organizational roles in Agile transformation process: A grounded theory approach. *Journal of Systems and Software*, 133(Supplement C):174–194.

[Juhola et al., 2014] Juhola, T., Yip, M. H., Hyrynsalmi, S., Mäkilä, T., and Leppänen, V. (2014). The Connection of the Stakeholder Cooperation Intensity and Team

Agility in Software Development. In *ICMIT '14: Proceedings of the 2014 International Conference on Management of Innovation and Technology*, pages 199–204. IEEE Computer Society.

[Juristo et al., 2006] Juristo, N., Moreno, A. M., and Strigel, W. (2006). Software Testing Practices in Industry. *IEEE Software*, 23(4):19–21.

[Kajava et al., 2006] Kajava, J., Anttila, J., Varonen, R., Savola, R., and Röning, J. (2006). Senior Executives Commitment to Information Security - from Motivation to Responsibility. In *CIS '06: Proceedings of the 2006 International Conference on Computational Intelligence and Security*, volume 2, pages 1519–1522. IEEE Computer Society.

[Kamsties and Lott, 1995] Kamsties, E. and Lott, C. M. (1995). An Empirical Evaluation of Three Defect-Detection Techniques. In *Proceedings of the 5th European Software Engineering Conference*, pages 362–383. Springer.

[Kaner et al., 2002] Kaner, C., Bach, J., and Pettichord, B. (2002). *Lessons Learned in Software Testing: A Context-Driven Approach*. John Wiley & Sons.

[Kangasniemi et al., 2015] Kangasniemi, M., Blomberg, K., and Korhonen, A. (2015). Mothers' perceptions of their health choices, related duties and responsibilities: A qualitative interview study. *Midwifery*, 31(11):1039–1044.

[Karhu et al., 2007] Karhu, K., Taipale, O., and Smolander, K. (2007). Outsourcing and Knowledge Management in Software Testing. In *EASE '07: Proceedings of the 11th International Conference on Evaluation and Assessment in Software Engineering*, pages 53–63. British Computer Society.

[Kasunic, 2005] Kasunic, M. (2005). Designing an Effective Survey. Technical report, Carnegie Mellon University CMU/SEI-2005-HB-004.

[Kasurinen et al., 2010] Kasurinen, J., Taipale, O., and Smolander, K. (2010). Software Test Automation in Practice: Empirical Observations. *Advances in Software Engineering*, 2010.

[Kautz and Jensen, 2013] Kautz, K. and Jensen, T. B. (2013). Sociomateriality at the royal court of IS: A jester's monologue. *Information and Organization*, 23(1):15–27.

[Kautz and Thaysen, 2001] Kautz, K. and Thaysen, K. (2001). Knowledge, learning and IT support in a small software company. *Journal of Knowledge Management*, 5(4):349–357.

[Kays et al., 2013] Kays, K. M., Keith, T. L., and Broughal, M. T. (2013). Best Practice in Online Survey Research with Sensitive Topics. In Sappleton, N., editor, *Advancing Research Methods with New Technologies*, pages 157–168. IGI Global.

[Kazemian and Howles, 2005] Kazemian, F. and Howles, T. (2005). A Software Testing Course for Computer Science Majors. *ACM SIGCSE Bulletin*, 37(4):50–53.

[Kelley et al., 2003] Kelley, K., Clark, B., Brown, V., and Sitzia, J. (2003). Good practice in the conduct and reporting of survey research. *International Journal for Quality in Health Care*, 15(3):261–266.

[Kelly and Shepard, 2002] Kelly, D. and Shepard, T. (2002). Qualitative Observations from Software Code Inspection Experiments. In *CASCON '02: Proceedings of the 2002 Conference of the Centre for Advanced Studies on Collaborative Research*, pages 35–46. IBM Press.

[Kettunen et al., 2010] Kettunen, V., Kasurinen, J., Taipale, O., and Smolander, K. (2010). A Study on Agility and Testing Processes in Software Organizations. In *ISSTA '10: Proceedings of the 19th International Symposium on Software Testing and Analysis*, pages 231–240. ACM Press.

[Khan and Parkinson, 2018] Khan, S. and Parkinson, S. (2018). Review into State of the Art of Vulnerability Assessment using Artificial Intelligence. In Parkinson, S., Crampton, A., and Hill, R., editors, *Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach*, pages 3–32. Springer.

[Kim, 2008] Kim, J.-H. (2008). *Faculty Self-Archiving Behavior: Factors Affecting the Decision to Self-Archive*. PhD thesis, University of Michigan.

[King, 2004] King, N. (2004). Using Templates in the Thematic Analysis of Text. In Cassell, C. and Symon, G., editors, *Essential Guide to Qualitative Methods in Organizational Research*, chapter 21, pages 256–270. SAGE Publications.

[Kitchenham and Charters, 2007] Kitchenham, B. and Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical report, Keele University and University of Durham EBSE-2007-01.

[Kitchenham and Pfleeger, 2002a] Kitchenham, B. and Pfleeger, S. L. (2002a). Principles of Survey Research Part 4: Questionnaire Evaluation. *ACM SIGSOFT Software Engineering Notes*, 27(3):20–23.

[Kitchenham and Pfleeger, 2002b] Kitchenham, B. and Pfleeger, S. L. (2002b). Principles of Survey Research: Part 5: Populations and Samples. *ACM SIGSOFT Software Engineering Notes*, 27(5):17–20.

[Kitchenham et al., 1995] Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case Studies for Method and Tool Evaluation. *IEEE Software*, 12(4):52–62.

[Kitchenham and Pfleeger, 2002c] Kitchenham, B. A. and Pfleeger, S. L. (2002c). Principles of Survey Research Part 2: Designing a Survey. *ACM SIGSOFT Software Engineering Notes*, 27(1):18–20.

[Kitchin and Dodge, 2011] Kitchin, R. and Dodge, M. (2011). *Code/Space: Software and Everyday Life*. MIT Press.

[Kizza, 2017] Kizza, J. M. (2017). *Guide to Computer Network Security*. Springer, fourth edition.

[Klein and Kleinman, 2002] Klein, H. K. and Kleinman, D. L. (2002). The Social Construction of Technology: Structural Considerations. *Science, Technology, & Human Values*, 27(1):28–52.

[Kling et al., 2003] Kling, R., McKim, G., and King, A. (2003). A Bit More to It: Scholarly Communication Forums as Socio-Technical Interaction Networks. *Journal of the American Society for Information Science and Technology*, 54(1):47–67.

[Knapp et al., 2006] Knapp, K. J., Marshall, T. E., Rainer, R. K., and Ford, F. N. (2006). Information security: management's effect on culture and policy. *Information Management & Computer Security*, 14(1):24–36.

[Kolarz, 2016] Kolarz, P. (2016). *Giddens and Politics beyond the Third Way: Utopian Realism in the Late Modern Age*. Palgrave Macmillan.

[Kollanus, 2005] Kollanus, S. (2005). ICMM - Inspection Capability Maturity Model. In *Proceedings of IASTED Conference on Software Engineering*, pages 372–377.

[Kollanus, 2009] Kollanus, S. (2009). Experiences from using ICMM in inspection process assessment. *Software Quality Journal*, 17(2):177–187.

[Kollanus, 2010] Kollanus, S. (2010). Test-Driven Development - Still a Promising Approach? In *QUATIC '10: Proceedings of the 7th International Conference on the Quality of Information and Communications Technology*, pages 403–408. IEEE Computer Society.

[Kollanus, 2011] Kollanus, S. (2011). ICMM - A Maturity Model for Software Inspections. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(5):327–341.

[Kollanus and Koskinen, 2006] Kollanus, S. and Koskinen, J. (2006). Software Inspections in Practice: Six Case Studies. In Münch, J. and Vierimaa, M., editors, *Product-Focused Software Process Improvement*, volume 4034 of *Lecture Notes in Computer Science*, pages 377–382. Springer.

[Kongsli, 2006] Kongsli, V. (2006). Towards Agile Security in Web Applications. In *OOPSLA '06: Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, pages 805–808. ACM Press.

[Kontio, 1998] Kontio, J. (1998). A Software Process Engineering Framework. In Zelkowitz, M. V., editor, *Advances in Computers: The Engineering of Large Systems*, chapter 2, pages 35–108. Academic Press.

[Koomen, 2002] Koomen, T. (2002). Worldwide Survey on Test Process Improvement. Technical report, Sogeti.

[Koomen and Pol, 1999] Koomen, T. and Pol, M. (1999). *Test Process Improvement: A practical step-by-step guide to structured testing.* Addison-Wesley.

[Kotulic and Clark, 2004] Kotulic, A. G. and Clark, J. G. (2004). Why there aren't more information security research studies. *Information & Management*, 41(5):597–607.

[Kreeger, 2009] Kreeger, M. N. (2009). Security Testing: Mind the Knowledge Gap. *ACM SIGCSE Bulletin*, 41(2):99–102.

[Kreeger and Harindranath, 2012] Kreeger, M. N. and Harindranath, G. (2012). Security Verification and Validation by Software SMEs: Theory versus Practice. In *Conf-IRM '12: Proceedings of the 2012 International Conference on Information Resources Management.* Vienna University of Economics and Business.

[Kreeger and Harindranath, 2017] Kreeger, M. N. and Harindranath, G. (2017). Security V&V Within Software SMEs: A Socio-Technical Interaction Network Analysis. In *Conf-IRM '17: Proceedings of the 2017 International Conference on Information Resources Management.* University of Chile.

[Kreeger and Lativy, 2008] Kreeger, M. N. and Lativy, N. (2008). Abandoning Proprietary Test Tools for Graphical User Interface Verification. In *TAIC-PART '08: Proceedings of the Testing: Academic & Industrial Conference - Practice and Research Techniques*, pages 52–56. IEEE Computer Society.

[Krein et al., 2014] Krein, J. L., Knutson, C. D., and Bird, C. (2014). Report from the 3rd International Workshop on Replication in Empirical Software Engineering Research (RESER 2013). *ACM SIGSOFT Software Engineering Notes*, 39(1):31–35.

[Krishna et al., 2004] Krishna, S., Sahay, S., and Walsham, G. (2004). Managing Cross-cultural Issues in Global Software Outsourcing. *Communications of the ACM*, 47(4):62–66.

[Kruger and Kearney, 2006] Kruger, H. A. and Kearney, W. D. (2006). A prototype for assessing information security awareness. *Computers & Security*, 25(4):289–296.

[Kuusisto and Ilvonen, 2003] Kuusisto, T. and Ilvonen, I. (2003). Information Security Culture in Small and Medium Size Enterprises. In *Frontiers of e-Business Research 2003*, pages 431–439. Tampere University of Technology and University of Tampere.

[Kvale, 1996] Kvale, S. (1996). *InterViews: An Introduction to Qualitative Research Interviewing*. SAGE Publications.

[Kwon and Johnson, 2014] Kwon, J. and Johnson, M. E. (2014). Proactive Versus Reactive Security Investments in the Healthcare Sector. *MIS Quarterly*, 38(2):451–471.

[Landis, 2011] Landis, S. (2011). *Agile Hiring*. Artima Press.

[Landman et al., 2017] Landman, D., Serebrenik, A., and Vinju, J. J. (2017). Challenges for Static Analysis of Java Reflection - Literature Review and Empirical Study. In *ICSE '17: Proceedings of the 39th International Conference on Software Engineering*, pages 507–518. IEEE Computer Society.

[Larusdottir et al., 2010] Larusdottir, M., Bjarnadottir, E., and Gulliksen, J. (2010). The Focus on Usability in Testing Practices in Industry. In Forbrig, P., Paternó, F., and Pejtersen, A. M., editors, *Human-Computer Interaction*, volume 332 of *IFIP Advances in Information and Communication Technology*, pages 98–109. Springer.

[Latour, 1987] Latour, B. (1987). *Science in Action: How to Follow Scientists and Engineers Through Society*. Harvard University Press.

[Latour and Woolgar, 1986] Latour, B. and Woolgar, S. (1986). *Laboratory Life: The Construction of Scientific Facts*. Princeton University Press.

[LaToza et al., 2006] LaToza, T. D., Venolia, G., and DeLine, R. (2006). Maintaining Mental Models: A Study of Developer Work Habits. In *ICSE '06: Proceedings of the 28th International Conference on Software Engineering*, pages 492–501. ACM Press.

[Law, 1992] Law, J. (1992). Notes on the Theory of the Actor-Network: Ordering, Strategy, and Heterogeneity. *Systemic Practice and Action Research*, 5(4):379–393.

[Leech, 2002] Leech, B. L. (2002). Asking Questions: Techniques for Semistructured Interviews. *PS: Political Science and Politics*, 35(4):665–668.

[Lenberg et al., 2015] Lenberg, P., Feldt, R., and Wallgren, L. G. (2015). Human Factors Related Challenges in Software Engineering - An Industrial Perspective. In *CHASE '15: Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 43–49. IEEE Computer Society.

[Leonardi, 2013] Leonardi, P. M. (2013). Theoretical foundations for the study of sociomateriality. *Information and Organization*, 23(2):59–76.

[Leska, 2004] Leska, C. (2004). Testing Across the Curriculum: Square One! *Journal of Computing Sciences in Colleges*, 19(5):163–169.

[Letch and Carroll, 2007] Letch, N. and Carroll, J. (2007). Integrated E-Government Systems: Unintended Impacts for those at the Margins. In *ECIS '07: Proceedings of the 15th European Conference on Information Systems*, pages 2197–2208. University of St. Gallen.

[Lethbridge, 2000] Lethbridge, T. C. (2000). What Knowledge Is Important to a Software Professional? *IEEE Computer*, 33(5):44–50.

[Lethbridge et al., 2007] Lethbridge, T. C., Díaz-Herrera, J., LeBlanc, Jr., R. J., and Thompson, J. B. (2007). Improving Software Practice Through Education: Challenges and Future Trends. In *FOSE '07: 2007 Future of Software Engineering*, pages 12–28. IEEE Computer Society.

[Linares et al., 2018] Linares, J., Melendez, K., Flores, L., and Dávila, A. (2018). Project Portfolio Management in Small Context in Software Industry: A Systematic Literature Review. In Mejia, J., Muñoz, M., Rocha, Á., Quiñonez, Y., and Calvo-Manzano, J., editors, *CIMPS '17: Proceedings of the 6th International Conference on Software Process Improvement*, pages 45–60. Springer.

[LinkedIn, 2020] LinkedIn (2020). LinkedIn. `https://www.linkedin.com/`. Retrieved: 16th of January, 2020.

[Lipner, 2000] Lipner, S. (2000). Security and Source Code Access: Issues and Realities. In *S&P 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 124–125. IEEE Computer Society.

[Mac an Bhaird, 2010] Mac an Bhaird, C. (2010). *Resourcing Small and Medium Sized Enterprises: A Financial Growth Life Cycle Approach*. Springer.

[Machado et al., 2010] Machado, P., Vincenzi, A., and Maldonado, J. (2010). Software Testing: An Overview. In Borba, P., Cavalcanti, A., Sampaio, A., and Woodcook, J., editors, *Testing Techniques in Software Engineering*, volume 6153 of *Lecture Notes in Computer Science*, pages 1–17. Springer.

[Mahmood et al., 2012] Mahmood, R., Esfahani, N., Kacem, T., Mirzaei, N., Malek, S., and Stavrou, A. (2012). A Whitebox Approach for Automated Security Testing of Android Applications on the Cloud. In *AST '12: Proceedings of the 2012 International Workshop on Automation of Software Test*, pages 22–28. IEEE Computer Society.

[Maier et al., 2017] Maier, P., Ma, Z., and Bloem, R. (2017). Towards a Secure SCRUM Process for Agile Web Application Development. In *ARES '17: Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM Press.

[Majchrzak and Jarvenpaa, 2004] Majchrzak, A. and Jarvenpaa, S. L. (2004). Information Security in Cross-Enterprise Collaborative Knowledge Work. *Emergence: Complexity and Organization*, 6(4):4–14.

[Mandić et al., 2009] Mandić, V., Markkula, J., and Oivo, M. (2009). Towards Multi-Method Research Approach in Empirical Software Engineering. In Bomarius, F., Oivo, M., Jaring, P., and Abrahamsson, P., editors, *Product-Focused Software Process Improvement*, volume 32 of *Lecture Notes in Business Information Processing*, pages 96–110. Springer.

[Mäntylä et al., 2015] Mäntylä, M. V., Adams, B., Khomh, F., Engström, E., and Petersen, K. (2015). On rapid releases and software testing: a case study and a semi-systematic literature review. *Empirical Software Engineering*, 20(5):1384–1425.

[Mäntylä et al., 2012] Mäntylä, M. V., Itkonen, J., and Iivonen, J. (2012). Who tested my software? Testing as an organizationally cross-cutting activity. *Software Quality Journal*, 20(1):145–172.

[Mäntylä and Lassenius, 2009] Mäntylä, M. V. and Lassenius, C. (2009). What Types of Defects Are Really Discovered in Code Reviews? *IEEE Transactions on Software Engineering*, 35(3):430–448.

[Marrero and Settle, 2005] Marrero, W. and Settle, A. (2005). Testing First: Emphasizing Testing in Early Programming Courses. *ACM SIGCSE Bulletin*, 37(3):4–8.

[Martin et al., 2007] Martin, D., Rooksby, J., Rouncefield, M., and Sommerville, I. (2007). 'Good' Organisational Reasons for 'Bad' Software Testing: An Ethnographic Study of Testing in a Small Software Company. In *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*, pages 602–611. IEEE Computer Society.

[Martin et al., 2008] Martin, D., Rooksby, J., Rouncefield, M., and Sommerville, I. (2008). Cooperative Work in Software Testing. In *CHASE '08: Proceedings of the 1st International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 93–96. ACM Press.

[Martin, 1999] Martin, W. (1999). The Social and Cultural Shaping of Educational Technology: Toward a Social Constructivist Framework. *AI & Society*, 13(4):402–420.

[Martínez-Caro and Cegarra-Navarro, 2010] Martínez-Caro, E. and Cegarra-Navarro, J. G. (2010). The impact of e-business on capital productivity: An analysis of the UK telecommunications sector. *International Journal of Operations & Production Management*, 30(5):488–507.

[Martínez-Ruiz et al., 2012] Martínez-Ruiz, T., Münch, J., García, F., and Piattini, M. (2012). Requirements and constructors for tailoring software processes: a systematic literature review. *Software Quality Journal*, 20(1):229–260.

[Mařík et al., 2000] Mařík, V., Král, L., and Mařík, R. (2000). Software Testing & Diagnostics: Theory & Practice. In *SOFSEM '00: Proceedings of the 27th Conference on Current Trends in Theory and Practice of Informatics*, pages 88–114. Springer.

[Mawson and Brown, 2017] Mawson, S. and Brown, R. (2017). Entrepreneurial acquisitions, open innovation and UK high growth SMEs. *Industry and Innovation*, 24(4):382–402.

[McCoy and Rosenbaum, 2017] McCoy, C. and Rosenbaum, H. (2017). Unintended and Shadow Practices of Decision Support System Dashboards in Higher Education

Institutions. In *ASIS&T '17: Proceedings of the Association for Information Science and Technology*, pages 757–758. John Wiley & Sons.

[McCoyd and Kerson, 2006] McCoyd, J. L. M. and Kerson, T. S. (2006). Conducting Intensive Interviews Using Email: A Serendipitous Comparative Opportunity. *Qualitative Social Work*, 5(3):389–406.

[McDermott, 2000] McDermott, J. P. (2000). Attack Net Penetration Testing. In *NSPW '00: Proceedings of the 2000 New Security Paradigms Workshop*, pages 15–21. ACM Press.

[McGraw, 2004] McGraw, G. (2004). Software Security. *IEEE Security & Privacy*, 2(2):80–83.

[McGraw, 2006] McGraw, G. (2006). *Software Security: Building Security In*. Addison-Wesley.

[McGraw, 2008] McGraw, G. (2008). Automated Code Review Tools for Security. *IEEE Computer*, 41(12):108–111.

[McGraw et al., 2016] McGraw, G., Migues, S., and West, J. (2016). Building Security In Maturity Model - Version 7. `https://www.synopsys.com/content/dam/synopsys/bsimm/reports/BSIMM7.pdf`. Retrieved: 16th of January, 2020.

[McLean and Hassard, 2004] McLean, C. and Hassard, J. (2004). Symmetrical Absence/Symmetrical Absurdity: Critical Notes on the Production of Actor-Network Accounts. *Journal of Management Studies*, 41(3):493–519.

[McManus, 2004] McManus, J. (2004). *Managing Stakeholders in Software Development Projects*. Routledge.

[McPhee et al., 2014] McPhee, R. D., Poole, M. S., and Iverson, J. (2014). Structuration Theory. In Putnam, L. L. and Mumby, D. K., editors, *The SAGE Handbook of Organizational Communication: Advances in Theory, Research, and Methods*, chapter 3, pages 75–100. SAGE Publications, third edition.

[Meehan and Richardson, 2002] Meehan, B. and Richardson, I. (2002). Identification of Software Process Knowledge Management. *Software Process: Improvement and Practice*, 7(2):47–55.

[Meier, 2006] Meier, J. D. (2006). Web Application Security Engineering. *IEEE Security & Privacy*, 4(4):16–24.

[Melian and Mähring, 2008] Melian, C. and Mähring, M. (2008). Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard. In Russo, B., Damiani, E., Hissam, S., Lundell, B., and Succi, G., editors, *Open Source Development, Communities and Quality*, volume 275 of *IFIP International Federation for Information Processing*, pages 93–104. Springer.

[Memon, 2002] Memon, A. M. (2002). GUI Testing: Pitfalls and Process. *IEEE Computer*, 35(8):87–88.

[Menezes et al., 1996] Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press.

[Mercuri, 2004] Mercuri, R. T. (2004). The HIPAA-potamus in Health Care Data Security. *Communications of the ACM*, 47(7):25–28.

[Messerschmitt and Szyperski, 2003] Messerschmitt, D. and Szyperski, C. (2003). *Software Ecosystem: Understanding an Indispensable Technology and Industry*. MIT Press.

[Meszaros and Aston, 2007] Meszaros, G. and Aston, J. (2007). Agile ERP: "You don't know what you've got 'till it's gone!". In *AGILE '07: Proceedings of the AGILE 2007 Conference*, pages 143–149. IEEE Computer Society.

[Meyer, 2006] Meyer, E. T. (2006). Socio-Technical Interaction Networks: A Discussion of the Strengths, Weaknesses and Future of Kling's STIN Model. In Berleur, J., Nurminen, M. I., and Impagliazzo, J., editors, *Social Informatics: An Information Society for all? In Remembrance of Rob Kling*, volume 223 of *IFIP International Federation for Information Processing*, pages 37–48. Springer.

[Meyer, 2007] Meyer, E. T. (2007). *Socio-Technical Perspectives on Digital Photography: Scientific Digital Photography Use by Marine Mammal Researchers*. PhD thesis, Indiana University.

[Meyer, 2014] Meyer, E. T. (2014). Examining the Hyphen: The Value of Social Informatics for Research and Teaching. In Fichman, P. and Rosenbaum, H., editors, *Social Informatics: Past, Present and Future*, chapter 3, pages 56–72. Cambridge Scholars Publishing.

[Meyer et al., 2011] Meyer, E. T., Thomas, A., and Schroeder, R. (2011). Web Archives: The Future(s). `https://ssrn.com/abstract=1830025`. Retrieved: 16th of January, 2020.

[Michael et al., 2013] Michael, C. C., van Wyk, K., and Radosevich, W. (2013). Risk-Based and Functional Security Testing. https://www.us-cert.gov/bsi/articles/best-practices/security-testing/risk-based-and-functional-security-testing. Retrieved: 16th of January, 2020.

[Microsoft, 2020] Microsoft (2020). Microsoft Security Development Lifecycle. https://www.microsoft.com/en-us/securityengineering/sdl/. Retrieved: 16th of January, 2020.

[Microsoft et al., 2008] Microsoft, BCS, and Intellect (2008). Developing the Future: Challenges and opportunities facing the UK Software Development Industry - Summary Findings and Recommendations. http://download.microsoft.com/documents/UK/developingthefuture/Developing_The_Future_08_Summary.pdf. Retrieved: 16th of January, 2020.

[Microsoft Patterns & Practices Team, 2009] Microsoft Patterns & Practices Team (2009). *Microsoft Application Architecture Guide*. Microsoft Press, second edition.

[Mingers, 2001] Mingers, J. (2001). Combining IS Research Methods: Towards a Pluralist Methodology. *Information Systems Research*, 12(3):240–259.

[Mishra and Mishra, 2013] Mishra, A. and Mishra, D. (2013). Software Project Management Tools: A Brief Comparative View. *ACM SIGSOFT Software Engineering Notes*, 38(3):1–4.

[Mitev, 2009] Mitev, N. (2009). In and out of actor-network theory: a necessary but insufficient journey. *Information Technology & People*, 22(1):9–25.

[Moe et al., 2014] Moe, N. B., Šmite, D., Hanssen, G. K., and Barney, H. (2014). From offshore outsourcing to insourcing and partnerships: four failed outsourcing attempts. *Empirical Software Engineering*, 19(5):1225–1258.

[Monteiro, 2001] Monteiro, E. (2001). Actor-Network Theory and Information Infrastructure. In Ciborra, C. U., editor, *From Control to Drift: The Dynamics of Corporate Information Infrastructures*, chapter 5, pages 71–83. Oxford University Press.

[Morgan, 2014] Morgan, D. L. (2014). *Integrating Qualitative & Quantitative Methods: A Pragmatic Approach*. SAGE Publications.

[Morisio et al., 2007] Morisio, M., Egorova, E., and Torchiano, M. (2007). Why software projects fail? Empirical evidence and relevant metrics. In *Proceedings of the*

International Conference on Software Process and Product Measurement, pages 299–308.

[Morris, 2009] Morris, A. (2009). Socio-Technical Systems in ICT: A Comprehensive Survey. Technical report, University of Trento DISI-09-054.

[Morrison, 2014] Morrison, J. G. (2014). *Understanding the Work of Telehealth Implementation Using Normalization Process Theory*. PhD thesis, Simon Fraser University.

[Mosley and Posey, 2002] Mosley, D. J. and Posey, B. A. (2002). *Just Enough Software Test Automation*. Prentice Hall.

[Mostashari, 2010] Mostashari, A. (2010). Sociotechnical Systems: A Conceptual Introduction. In Hu, F., Mostashari, A., and Xie, J., editors, *Socio-Technical Networks: Science and Engineering Design*, chapter 1, pages 1–11. CRC Press.

[Mueller et al., 2016] Mueller, B., Renken, U., and van den Heuvel, G. (2016). Get Your Act Together - An Alternative Approach to Understanding the Impact of Technology on Individual and Organizational Behavior. *The Data Base for Advances in Information Systems*, 47(4):67–83.

[Murphy et al., 2013] Murphy, B., Bird, C., Zimmermann, T., Williams, L., Nagappan, N., and Begel, A. (2013). Have Agile Techniques been the Silver Bullet for Software Development at Microsoft? In *ESEM '13: Proceedings of the 7th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 75–84. IEEE Computer Society.

[Murrill, 1998] Murrill, B. W. (1998). Integrating Software Analysis, Testing, and Verification into the Undergraduate Computer Science Curriculum. *Computer Science Education*, 8(2):85–99.

[Murthy, 2009] Murthy, U. S. (2009). Conducting Creativity Brainstorming Sessions in Small and Medium-Sized Enterprises Using Computer-Mediated Communication Tools. In Dhillon, G., Stahl, B. C., and Baskerville, R., editors, *Information Systems - Creativity and Innovation in Small and Medium-Sized Enterprises*, volume 301 of *IFIP Advances in Information and Communication Technology*, pages 42–59. Springer.

[Murugesan, 1994] Murugesan, S. (1994). Attitude Towards Testing: A Key Contributor to Software Quality. In *Proceedings of the 1st International Conference on*

Software Testing, Reliability and Quality Assurance, pages 111–115. IEEE Computer Society.

[Mutafelija and Stromberg, 2003] Mutafelija, B. and Stromberg, H. (2003). *Systematic Process Improvement Using ISO 9001:2000 and CMMI®*. Artech House.

[Mutch, 2013] Mutch, A. (2013). Sociomateriality - Taking the wrong turning? *Information and Organization*, 23(1):28–40.

[Mwenya and Brown, 2017] Mwenya, J. K. and Brown, I. (2017). Actor-Network Theory in IS Research: Critique on Application of the Principle of Generalized Symmetry. In *SAICSIT '17: Proceedings of the 2017 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 24:1–24:10. ACM Press.

[Myers, 1978] Myers, G. J. (1978). A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections. *Communications of the ACM*, 21(9):760–768.

[Myers, 2004] Myers, G. J. (2004). *The Art of Software Testing*. John Wiley & Sons, second edition.

[Myers et al., 2011] Myers, G. J., Sandler, C., and Badgett, T. (2011). *The Art of Software Testing*. John Wiley & Sons, third edition.

[Nagy and Víg, 2008] Nagy, T. and Víg, A. N. (2008). Erlang Testing and Tools Survey. In *ERLANG '08: Proceedings of the 7th ACM SIGPLAN workshop on ERLANG*, pages 21–28. ACM Press.

[Nakagawa et al., 2007] Nakagawa, E. Y., da Silva Simão, A., Ferrari, F. C., and Maldonado, J. C. (2007). Towards a Reference Architecture for Software Testing Tools. In *SEKE '07: Proceedings of the 19th International Conference on Software Engineering & Knowledge Engineering*, pages 157–162.

[National Research Council, 1996] National Research Council (1996). *Statistical Software Engineering*. The National Academies Press.

[National Research Council, 2004] National Research Council (2004). *An Assessment of the Small Business Innovation Research Program: Project Methodology*. The National Academies Press.

[Nejmeh and Riddle, 2006] Nejmeh, B. A. and Riddle, W. E. (2006). The PERFECT Approach to Experience-Based Process Evolution. In Zelkowitz, M. V., editor, *Ad-*

vances in Computers: Quality Software Development, chapter 5, pages 173–238. Academic Press.

[Nelson and Quick, 2013] Nelson, D. L. and Quick, J. C. (2013). *Organizational Behavior: Science, the Real World, and You*. Cengage Learning, eighth edition.

[Nelson and Schumann, 2004] Nelson, S. and Schumann, J. (2004). What makes a Code Review Trustworthy? In *HICSS '04: Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society.

[Nerur et al., 2005] Nerur, S., Mahapatra, R., and Mangalaraj, G. (2005). Challenges of Migrating to Agile Methodologies. *Communications of the ACM*, 48(5):72–78.

[Neuhaus et al., 2007] Neuhaus, S., Zimmermann, T., Holler, C., and Zeller, A. (2007). Predicting Vulnerable Software Components. In *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 529–540. ACM Press.

[Neumann, 1995] Neumann, P. G. (1995). *Computer-Related Risks*. ACM Press/Addison-Wesley.

[Ngo et al., 2009] Ngo, L., Zhou, W., Chonka, A., and Singh, J. (2009). Assessing the Level of I.T. Security Culture Improvement: Results from Three Australian SMEs. In *Proceedings of the 35th Annual Conference of the IEEE Industrial Electronics Society*, pages 3189–3195. IEEE Computer Society.

[Nguyen et al., 2006] Nguyen, H. Q., Hackett, M., and Whitlock, B. K. (2006). *Happy About® Global Software Test Automation: A Discussion of Software Testing for Executives*. Happy About.

[Nicholson and Sahay, 2001] Nicholson, B. and Sahay, S. (2001). Some political and cultural issues in the globalisation of software development: case experience from Britain and India. *Information and Organization*, 11(1):25–43.

[Niemi, 2007] Niemi, E. (2007). Enterprise Architecture Stakeholders - A Holistic View. In *AMCIS '07: Proceedings of the Thirteenth Americas Conference on Information Systems*, number 41.

[Nimmo, 2011] Nimmo, R. (2011). Actor-Network Theory and Methodology: Social Research in a More-Than-Human World. *Methodological Innovations Online*, 6(3):108–119.

[NIST, 2001] NIST (2001). FIPS PUB 140-2: Security Requirements for Cryptographic Modules. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf. Retrieved: 16th of January, 2020.

[NIST, 2002] NIST (2002). The Economic Impacts of Inadequate Infrastructure for Software Testing. https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf. Retrieved: 16th of January, 2020.

[NIST, 2014] NIST (2014). Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53Ar4.pdf. Retrieved: 16th of January, 2020.

[NIST, 2020] NIST (2020). National Vulnerability Database - Search and Statistics. https://nvd.nist.gov/vuln/search. Retrieved: 16th of January, 2020.

[Novelli and Wenzel, 2013] Novelli, F. and Wenzel, S. (2013). Online and Offline Sales Channels for Enterprise Software: Cannibalization or Complementarity? In *ICIS '13: Proceedings of the 34th International Conference on Information Systems*.

[Nowell et al., 2017] Nowell, L. S., Norris, J. M., White, D. E., and Moules, N. J. (2017). Thematic Analysis: Striving to Meet the Trustworthiness Criteria. *International Journal of Qualitative Methods*, 16(1):1–13.

[OECD, 2002] OECD (2002). OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security. http://www.oecd.org/sti/ieconomy/15582260.pdf. Retrieved: 16th of January, 2020.

[Office for National Statistics, 2009] Office for National Statistics (2009). UK Standard Industrial Classification of Economic Activities 2007 (SIC 2007). http://www.ons.gov.uk/ons/guide-method/classifications/current-standard-classifications/standard-industrial-classification/sic2007---explanatory-notes.pdf. Retrieved: 16th of January, 2020.

[Offutt, 2002] Offutt, J. (2002). Quality Attributes of Web Software Applications. *IEEE Software*, 19(2):25–32.

[Okere et al., 2012] Okere, I., van Niekerk, J., and Carroll, M. (2012). Assessing Information Security Culture: A Critical Analysis of Current Approaches. In *Proceedings of the Information Security for South Africa 2012 Conference*. IEEE Computer Society.

[Onwuegbuzie and Leech, 2005] Onwuegbuzie, A. J. and Leech, N. L. (2005). On Becoming a Pragmatic Researcher: The Importance of Combining Quantitative and Qualitative Research Methodologies. *International Journal of Social Research Methodology*, 8(5):375–387.

[O'Regan, 2011] O'Regan, G. (2011). *Introduction to Software Process Improvement*. Springer.

[Orlikowski, 1992] Orlikowski, W. J. (1992). The Duality of Technology: Rethinking the Concept of Technology in Organizations. *Organization Science*, 3(3):398–427.

[Orlikowski, 2007] Orlikowski, W. J. (2007). Sociomaterial Practices: Exploring Technology at Work. *Organization Studies*, 28(9):1435–1448.

[Orlikowski, 2010] Orlikowski, W. J. (2010). The sociomateriality of organisational life: considering technology in management research. *Cambridge Journal of Economics*, 34(1):125–141.

[Orlikowski and Iacono, 2001] Orlikowski, W. J. and Iacono, C. S. (2001). Research Commentary: Desperately Seeking the "IT" in IT Research - A Call to Theorizing the IT Artifact. *Information Systems Research*, 12(2):121–134.

[Orlikowski and Scott, 2008] Orlikowski, W. J. and Scott, S. V. (2008). Sociomateriality: Challenging the Separation of Technology, Work and Organization. *The Academy of Management Annals*, 2(1):433–474.

[Osterweil, 1996] Osterweil, L. (1996). Strategic Directions in Software Quality. *ACM Computing Surveys*, 28(4):738–750.

[Overhage et al., 2011] Overhage, S., Schlauderer, S., Birkmeier, D., and Miller, J. (2011). What Makes IT Personnel Adopt Scrum? A Framework of Drivers and Inhibitors to Developer Acceptance. In *HICSS '11: Proceedings of the 44th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society.

[OWASP, 2017] OWASP (2017). Software Assurance Maturity Model: A guide to building security into software development - Version 1.5. `https://github.com/OWASP/samm/raw/master/Supporting%20Resources/v1.5/Final/SAMM_Core_V1-5_FINAL.pdf`. Retrieved: 16th of January, 2020.

[Owsley, 2017] Owsley, B. L. (2017). Can Apple build a privacy minded iPhone security system so secure that Apple cannot access it? *Health and Technology*, 7(4):369–376.

[Oza et al., 2004] Oza, N., Hall, T., Rainer, A., and Grey, S. (2004). Critical Factors in Software Outsourcing - A Pilot Study. In *WISER '04: Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research*, pages 67–71. ACM Press.

[Oza and Hall, 2005] Oza, N. V. and Hall, T. (2005). Difficulties in Managing Offshore Software Outsourcing Relationships: An Empirical Analysis of 18 High Maturity Indian Software Companies. *Journal of Information Technology Case and Application Research*, 7(3):25–41.

[Oza et al., 2006] Oza, N. V., Hall, T., Rainer, A., and Grey, S. (2006). Trust in software outsourcing relationships: An empirical investigation of Indian software companies. *Information and Software Technology*, 48(5):345–354.

[Ozment, 2006] Ozment, A. (2006). Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models. In Gollmann, D., Massacci, F., and Yautsiukhin, A., editors, *Quality of Protection: Security Measurements and Metrics*, pages 25–36. Springer.

[Ozment, 2007] Ozment, A. (2007). *Vulnerability Discovery & Software Security*. PhD thesis, University of Cambridge.

[Page et al., 2008] Page, A., Johnston, K., and Rollison, B. (2008). *How We Test Software at Microsoft*. Microsoft Press.

[Pan, 2005] Pan, G. S. C. (2005). Information systems project abandonment: a stakeholder analysis. *International Journal of Information Management*, 25(2):173–184.

[Pan and Flynn, 2003] Pan, G. S. C. and Flynn, D. (2003). Towards a Stakeholder Analysis of Information Systems Development Project Abandonment. In *ECIS '03: Proceedings of the 11th European Conference on Information Systems*, pages 1462–1473.

[Parizi et al., 2018] Parizi, R. M., Dehghantanha, A., Choo, K.-K. R., and Singh, A. (2018). Empirical Vulnerability Analysis of Automated Smart Contracts Security Testing on Blockchains. In *CASCON '18: Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, pages 103–113. IBM Press.

[Park et al., 2008] Park, J., Ryu, H., Choi, H.-J., and Ryu, D.-K. (2008). A Survey on Software Test Maturity in Korean Defense Industry. In *ISEC '08: Proceedings of the 1st India Software Engineering Conference*, pages 149–150. ACM Press.

[Parmiggiani and Mikalsen, 2013] Parmiggiani, E. and Mikalsen, M. (2013). The Facets of Sociomateriality: A Systematic Mapping of Emerging Concepts and Definitions. In Aanestad, M. and Bratteteig, T., editors, *SCIS '13: Proceedings of the 4th Scandinavian Conference on Information Systems*, pages 87–103. Springer.

[Parnas and Lawford, 2003] Parnas, D. L. and Lawford, M. (2003). Inspection's Role in Software Quality Assurance. *IEEE Software*, 20(4):16–20.

[Pashchenko et al., 2017] Pashchenko, I., Dashevskyi, S., and Massacci, F. (2017). Delta-bench: Differential Benchmark for Static Analysis Security Testing Tools. In *ESEM '17: Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 163–168. IEEE Computer Society.

[Patton, 2005] Patton, R. (2005). *Software Testing*. Sams Publishing, second edition.

[Paulk, 2004] Paulk, M. C. (2004). Surviving the Quagmire of Process Models, Integrated Models, and Standards. In *Proceedings of the ASQ Annual Quality Congress*.

[Perry et al., 2000] Perry, D. E., Porter, A. A., and Votta, L. G. (2000). Empirical Studies of Software Engineering: A Roadmap. In *ICSE '00: Proceedings of the 22nd International Conference on Software Engineering - Future of Software Engineering Track*, pages 345–355. ACM Press.

[Perry, 2006] Perry, W. E. (2006). *Effective Methods for Software Testing*. John Wiley & Sons, third edition.

[Peters et al., 2017] Peters, F., Tun, T. T., Yu, Y., and Nuseibeh, B. (2017). Text Filtering and Ranking for Security Bug Report Prediction. *IEEE Transactions on Software Engineering*.

[Peters, 2003] Peters, L. (2003). Educating Software Engineering Managers. In *CSEET '03: Proceedings of the 16th Conference on Software Engineering Education and Training*, pages 78–85. IEEE Computer Society.

[Petersen et al., 2008] Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. In *EASE '08: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, pages 68–77. British Computer Society.

[Petersen and Gencel, 2013] Petersen, K. and Gencel, C. (2013). Worldviews, Research Methods, and their Relationship to Validity in Empirical Software Engineering Research. In *IWSM-Mensura '13: Proceedings of the 2013 Joint Conference of the 23rd*

International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, pages 81–89. IEEE Computer Society.

[Pettichord, 2000] Pettichord, B. (2000). Testers and Developers Think Differently: Understanding and utilizing the diverse traits of key players on your team. *Testing & Quality*, January/February:42–46.

[Pfahl et al., 2014] Pfahl, D., Yin, H., Mäntylä, M. V., and Münch, J. (2014). How is Exploratory Testing Used? A State-of-the-Practice Survey. In *ESEM '14: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM Press.

[Pfleeger and Pfleeger, 2012] Pfleeger, C. P. and Pfleeger, S. L. (2012). *Analyzing Computer Security: A Threat/Vulnerability/Countermeasure Approach*. Prentice Hall.

[Pfleeger and Kitchenham, 2001] Pfleeger, S. L. and Kitchenham, B. A. (2001). Principles of Survey Research Part 1: Turning Lemons into Lemonade. *ACM SIGSOFT Software Engineering Notes*, 26(6):16–18.

[Pham et al., 2017] Pham, R., Kiesling, S., Singer, L., and Schneider, K. (2017). Onboarding inexperienced developers: struggles and perceptions regarding automated testing. *Software Quality Journal*, 25(4):1239–1268.

[Phillips et al., 2003] Phillips, R., Freeman, R. E., and Wicks, A. C. (2003). What Stakeholder Theory Is Not. *Business Ethics Quarterly*, 13(4):479–502.

[Phillips, 1997] Phillips, R. A. (1997). Stakeholder Theory and A Principle of Fairness. *Business Ethics Quarterly*, 7(1):51–66.

[Piessens, 2002] Piessens, F. (2002). A Taxonomy of Causes of Software Vulnerabilities in Internet Software. In *Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering*, pages 47–52.

[Pikkarainen et al., 2012] Pikkarainen, M., Salo, O., Kuusela, R., and Abrahamsson, P. (2012). Strengths and barriers behind the successful agile deployment - insights from the three software intensive companies in Finland. *Empirical Software Engineering*, 17(6):675–702.

[Pinch, 1996] Pinch, T. J. (1996). The Social Construction of Technology: A Review. In Fox, R., editor, *Technological Change: Methods and Themes in the History of Technology*, chapter 1, pages 17–36. Harwood Academic Publishers.

[Pinch and Bijker, 1989] Pinch, T. J. and Bijker, W. E. (1989). The Social Construction of Facts and Artifacts: Or How the Sociology of Science and the Sociology of Technology Might Benefit Each Other. In Bijker, W. E., Hughes, T. P., and Pinch, T. J., editors, *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*, pages 17–50. MIT Press.

[Pope et al., 2000] Pope, C., Ziebland, S., and Mays, N. (2000). Analysing qualitative data. *The BMJ*, 320(7227):114–116.

[Porter et al., 1998] Porter, A., Siy, H., Mockus, A., and Votta, L. (1998). Understanding the Sources of Variation in Software Inspections. *ACM Transactions on Software Engineering and Methodology*, 7(1):41–79.

[Porter, 2004] Porter, S. R. (2004). Raising Response Rates: What Works? *New Directions for Institutional Research*, 2004(121):5–21.

[Poston and Sexton, 1992] Poston, R. M. and Sexton, M. P. (1992). Evaluating and Selecting Testing Tools. *IEEE Software*, 9(3):33–42.

[Potter and McGraw, 2004] Potter, B. and McGraw, G. (2004). Software Security Testing. *IEEE Security & Privacy*, 2(5):81–85.

[Pouloudi and Reed, 1998] Pouloudi, A. and Reed, C. (1998). Towards a Multi-Agent Representation of Stakeholder Interests. In *PAAM '98: Proceedings of the 3rd International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 393–407. Practical Application Company.

[Power, 2010a] Power, K. (2010a). Stakeholder Identification in Agile Software Product Development Organizations: A model for understanding who and what really counts. In *Agile '10: Proceedings of the 2010 Agile Conference*, pages 87–94. IEEE Computer Society.

[Power, 2010b] Power, K. (2010b). Stakeholder Theory and Agile Software Development. In Babar, M. A., Vierimaa, M., and Oivo, M., editors, *PROFES '10: Proceedings of the 11th International Conference on Product-Focused Software Process Improvement*, pages 97–98. ACM Press.

[Prell, 2009] Prell, C. (2009). Rethinking the Social Construction of Technology through 'Following the Actors': A Reappraisal of Technological Frames. *Sociological Research Online*, 14(2).

[Presser et al., 2004] Presser, S., Couper, M. P., Lessler, J. T., Martin, E., Martin, J., Rothgeb, J. M., and Singer, E. (2004). Methods for Testing and Evaluating Survey Questions. *The Public Opinion Quarterly*, 68(1):109–130.

[Puleio, 2006] Puleio, M. (2006). How Not to do Agile Testing. In *AGILE '06: Proceedings of the AGILE 2006 Conference*, pages 305–314. IEEE Computer Society.

[Punter et al., 2003] Punter, T., Ciolkowski, M., Freimut, B., and John, I. (2003). Conducting On-line Surveys in Software Engineering. In *ISESE '03: Proceedings of the 2nd ACM/IEEE International Symposium on Empirical Software Engineering*, pages 80–88. IEEE Computer Society.

[Pyhäjärvi et al., 2003] Pyhäjärvi, M., Rautiainen, K., and Itkonen, J. (2003). Increasing Understanding of the Modern Testing Perspective in Software Product Development Projects. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 250–259. IEEE Computer Society.

[Qu and Dumay, 2011] Qu, S. Q. and Dumay, J. (2011). The qualitative research interview. *Qualitative Research in Accounting & Management*, 8(3):238–264.

[Qualtrics, 2020] Qualtrics (2020). Qualtrics. `https://www.qualtrics.com/uk/`. Retrieved: 16th of January, 2020.

[Rainys, 2006] Rainys, R. (2006). How Can NRA Contribute to the Improvement of IT Security? In *ISSE 2006: Securing Electronic Busines Processes*, pages 426–432. Vieweg.

[Ramdani et al., 2013] Ramdani, B., Chevers, D., and Williams, D. A. (2013). SMEs' adoption of enterprise applications: A technology-organisation-environment model. *Journal of Small Business and Enterprise Development*, 20(4):735–753.

[Ramler et al., 2006] Ramler, R., Biffl, S., and Grünbacher, P. (2006). Value-Based Management of Software Testing. In Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., and Grünbacher, P., editors, *Value-Based Software Engineering*, pages 225–244. Springer.

[Ramler and Wolfmaier, 2006] Ramler, R. and Wolfmaier, K. (2006). Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost. In *AST '06: Proceedings of the 2006 International Workshop on Automation of Software Test*, pages 85–91. ACM Press.

[Ranganathan and Balaji, 2007] Ranganathan, C. and Balaji, S. (2007). Critical Capabilities for Offshore Outsourcing of Information Systems. *MIS Quarterly Executive*, 6(3):147–164.

[Rankin, 2002] Rankin, C. (2002). The Software Testing Automation Framework. *IBM Systems Journal*, 41(1):126–139.

[Ransome and Misra, 2014] Ransome, J. and Misra, A. (2014). *Core Software Security: Security at the Source*. CRC Press.

[Rashid et al., 2018] Rashid, M., Clarke, P. M., and O'Connor, R. V. (2018). An Approach to Investigating Proactive Knowledge Retention in OSS Communities. In *EuroSPI '18: Proceedings of the 25th European Conference on Systems, Software and Services Process Improvement*, pages 108–119. Springer.

[Rasmussen and Thimm, 2009] Rasmussen, K. B. and Thimm, H. (2009). Fact-Based Understanding of Business Survey Non-Response. *The Electronic Journal of Business Research Methods*, 7(1):83–92.

[Rautiainen et al., 2002] Rautiainen, K., Lassenius, C., and Sulonen, R. (2002). 4CC: A Framework for Managing Software Product Development. *Engineering Management Journal*, 14(2):27–32.

[Reece and Stahl, 2015] Reece, R. P. and Stahl, B. C. (2015). The professionalisation of information security: Perspectives of UK practitioners. *Computers & Security*, 48:182–195.

[Reinert, 2009] Reinert, M. E. (2009). *Studying the Development Conceptions of a New Company and Associated Information System Using the STIN Strategy*. PhD thesis, Pennsylvania State University.

[Rhodes, 2009] Rhodes, J. (2009). Using Actor-Network Theory to Trace an ICT (Telecenter) Implementation Trajectory in an African Women's Micro-Enterprise Development Organization. *Information Technologies & International Development*, 5(3):1–20.

[RHUL, 2020] RHUL (2020). LibrarySearch, Royal Holloway, University of London. `https://librarysearch.royalholloway.ac.uk/`. Retrieved: 16th of January, 2020.

[Rice, 2007] Rice, D. (2007). *Geekonomics: The Real Cost of Insecure Software*. Pearson Education.

[Rice and Bucholz, 2007] Rice, D. C. and Bucholz, G. (2007). Methods of Auditing Applications. In Tipton, H. F. and Krause, M., editors, *Information Security Management Handbook*, pages 2537–2545. CRC Press, sixth edition.

[Rice, 2016] Rice, R. W. (2016). The Elusive Tester-to-Developer Ratio. *Software Quality Professional*, 18(2):20–24.

[Riemer and Vehring, 2010] Riemer, K. and Vehring, N. (2010). It's not a property! Exploring the Sociomateriality of Software Usability. In *ICIS '10: Proceedings of the 31st International Conference on Information Systems*.

[Rigby et al., 2016] Rigby, D. K., Sutherland, J., and Takeuchi, H. (2016). Embracing Agile: How to Master the Process That's Transforming Management. *Harvard Business Review*, 94(5):40–50.

[Rindell et al., 2015] Rindell, K., Hyrynsalmi, S., and Leppänen, V. (2015). A Comparison of Security Assurance Support of Agile Software Development Methods. In *CompSysTech '15: Proceedings of the 16th International Conference on Computer Systems and Technologies*, pages 61–68. ACM Press.

[Rindell et al., 2017] Rindell, K., Hyrynsalmi, S., and Leppänen, V. (2017). Busting a Myth: Review of Agile Security Engineering Methods. In *ARES '17: Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM Press.

[Riungu-Kalliosaari et al., 2016] Riungu-Kalliosaari, L., Taipale, O., Smolander, K., and Richardson, I. (2016). Adoption and use of cloud-based testing in practice. *Software Quality Journal*, 24(2):337–364.

[Rob, 2003] Rob, M. A. (2003). Project Failures in Small Companies. *IEEE Software*, 20(6):94–95.

[Robbin, 2007] Robbin, A. (2007). Rob Kling In Search of One Good Theory. *The Information Society*, 23(4):235–250.

[Rodrigues et al., 2010] Rodrigues, A., Bessa, A., and Pinheiro, P. R. (2010). Barriers to Implement Test Process in Small-Sized Companies. In Lytras, M. D., Ordonez De Pablos, P., Ziderman, A., Roulstone, A., Maurer, H., and Imber, J. B., editors, *Organizational, Business, and Technological Aspects of the Knowledge Society*, volume 112, pages 233–242. Springer.

[Rogers et al., 2007] Rogers, M., Helmers, C., and Greenhalgh, C. (2007). An analysis of the characteristics of small and medium enterprises that use intellectual property. https://webarchive.nationalarchives.gov.uk/20140603115023/http://www.ipo.gov.uk/ipresearch-characteristics-200710.pdf. Retrieved: 16th of January, 2020.

[Rombach and Achatz, 2007] Rombach, D. and Achatz, R. (2007). Research Collaborations between Academia and Industry. In *FOSE '07: 2007 Future of Software Engineering*, pages 29–36. IEEE Computer Society.

[Rombach et al., 2008] Rombach, D., Ciolkowski, M., Jeffery, R., Laitenberger, O., McGarry, F., and Shull, F. (2008). Impact of Research on Practice in the field of Inspections, Reviews and Walkthroughs: Learning from Successful Industrial Uses. *ACM SIGSOFT Software Engineering Notes*, 33(6):26–35.

[Rooksby et al., 2009] Rooksby, J., Rouncefield, M., and Sommerville, I. (2009). Testing in the Wild: The Social and Organisational Dimensions of Real World Practice. *Computer Supported Cooperative Work*, 18(5-6):559–580.

[Root and Sweeney, 2006] Root, R. and Sweeney, M. R. (2006). *A Tester's Guide to .NET Programming*. Apress.

[Ropohl, 1999] Ropohl, G. (1999). Philosophy of Socio-Technical Systems. *Society for Philosophy and Technology*, 4(3):59–71.

[Rose, 1998] Rose, J. (1998). Evaluating the Contribution of Structuration Theory to the Information Systems Discipline. In *ECIS '98: Proceedings of the 6th European Conference on Information Systems*, pages 910–924.

[Rose and Scheepers, 2001] Rose, J. and Scheepers, R. (2001). Structuration Theory and Information System Development - Frameworks for Practice. In *ECIS '01: Proceedings of the 9th European Conference on Information Systems*, pages 217–231.

[Rosenbaum and Joung, 2004] Rosenbaum, H. and Joung, K. H. (2004). Socio-Technical Interaction Networks as a Tool for Understanding Digital Libraries. In *ASIS&T '04: Proceedings of the American Society for Information Science and Technology*, pages 206–212. John Wiley & Sons.

[Rossman and Wilson, 1985] Rossman, G. B. and Wilson, B. L. (1985). Numbers and Words: Combining Quantitative and Qualitative Methods in a Single Large-Scale Evaluation Study. *Evaluation Review*, 9(5):627–643.

[Rowley, 2012] Rowley, J. (2012). Conducting research interviews. *Management Research Review*, 35(3/4):260–271.

[Ruighaver et al., 2007] Ruighaver, A. B., Maynard, S. B., and Chang, S. (2007). Organisational security culture: Extending the end-user perspective. *Computers & Security*, 26(1):56–62.

[Runeson, 2006] Runeson, P. (2006). A Survey of Unit Testing Practices. *IEEE Software*, 23(4):22–29.

[Runeson and Höst, 2009] Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164.

[Rungi and Matulevičius, 2013] Rungi, K. and Matulevičius, R. (2013). Empirical Analysis of the Test Maturity Model Integration (TMMi). In Skersys, T., Butleris, R., and Butkiene, R., editors, *ICIST '13: Proceedings of the 19th International Conference on Information and Software Technologies*, pages 376–391. Springer.

[Rus and Lindvall, 2002] Rus, I. and Lindvall, M. (2002). Knowledge Management in Software Engineering. *IEEE Software*, 19(3):26–38.

[Russell and Van Duren, 2016] Russell, B. and Van Duren, D. (2016). *Practical Internet of Things Security*. Packt Publishing.

[Russell, 1986] Russell, S. (1986). The Social Construction of Artefacts: A Response to Pinch and Bijker. *Social Studies of Science*, 16(2):331–346.

[Russo et al., 2017] Russo, D., Ciancarini, P., Falasconi, T., and Tomasi, M. (2017). Software Quality Concerns in the Italian Bank Sector: The Emergence of a Meta-Quality Dimension. In *ICSE-SEIP '17: Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track*, pages 63–72. IEEE Computer Society.

[Russo et al., 2018] Russo, D., Ciancarini, P., Falasconi, T., and Tomasi, M. (2018). A Meta-Model for Information Systems Quality: A Mixed Study of the Financial Sector. *ACM Transactions on Management Information Systems*, 9(3):11:1–11:38.

[Ryan and O'Connor, 2013] Ryan, S. and O'Connor, R. V. (2013). Acquiring and sharing tacit knowledge in software development teams: An empirical study. *Information and Software Technology*, 55(9):1614–1624.

[Sambrook, 2005] Sambrook, S. (2005). Factors Influencing the Context and Process of Work-Related Learning: Synthesizing Findings from Two Research Projects. *Human Resource Development International*, 8(1):101–119.

[Sánchez et al., 2010] Sánchez, L. E., Santos-Olmo, A., Fernández-Medina, E., and Piattini, M. (2010). Security Culture in Small and Medium-Size Enterprise. In Varajão, J. E. Q., Cruz-Cunha, M. M., Putnik, G. D., and Trigo, A., editors, *ENTERprise Information Systems*, volume 110, pages 315–324. Springer.

[Sánchez-Gordón et al., 2017] Sánchez-Gordón, M.-L., Colomo-Palacios, R., Sánchez, A., de Amescua Seco, A., and Larrucea, X. (2017). Towards the Integration of Security Practices in the Software Implementation Process of ISO/IEC 29110: A Mapping. In Stolfa, J., Stolfa, S., O'Connor, R. V., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, pages 3–14. Springer.

[Sandelin and Vierimaa, 2003] Sandelin, T. and Vierimaa, M. (2003). Empirical Studies in ESERNET. In Conradi, R. and Wang, A. I., editors, *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, volume 2765, pages 39–54. Springer.

[Santos et al., 2017] Santos, R. E. S., Magalhães, C. V. C., Correia-Neto, J. S., da Silva, F. Q. B., Capretz, L. F., and Souza, R. E. C. (2017). Would You Like to Motivate Software Testers? Ask Them How. In *ESEM '17: Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 95–104. IEEE Computer Society.

[Sawyer, 2001] Sawyer, S. (2001). Effects of intra-group conflict on packaged software development team performance. *Information Systems Journal*, 11(2):155–178.

[Sawyer, 2004] Sawyer, S. (2004). Software Development Teams. *Communications of the ACM*, 47(12):95–99.

[Sayes, 2017] Sayes, E. (2017). Marx and the critique of Actor-Network Theory: mediation, translation, and explanation. *Distinktion: Journal of Social Theory*, 18(3):294–313.

[Scacchi, 2005] Scacchi, W. (2005). Socio-Technical Interaction Networks in Free/Open Source Software Development Processes. In Acuña, S. T. and Juristo, N., editors, *Software Process Modeling*, volume 10 of *International Series in Software Engineering*, pages 1–27. Springer.

[Schechter, 2002] Schechter, S. (2002). How to Buy Better Testing Using Competition to Get the Most Security and Robustness for Your Dollar. In Davida, G., Frankel, Y., and Rees, O., editors, *InfraSec '02: Proceedings of the International Conference on Infrastructure Security*, pages 73–87. Springer.

[Schleife et al., 2017] Schleife, K., Niemann, F., Dupuis, D., Beckert, B., and Wydra, S. (2017). The Economic and Social Impact of Software & Services on Competitiveness and Innovation. `http://ec.europa.eu/newsroom/dae/document.cfm?doc_id=43825`. Retrieved: 16th of January, 2020.

[Schlienger and Teufel, 2003] Schlienger, T. and Teufel, S. (2003). Information security culture - from analysis to change. *South African Computer Journal*, 31:46–52.

[Schneier, 2003] Schneier, B. (2003). *Beyond Fear: Thinking Sensibly About Security in an Uncertain World*. Springer.

[Schneier, 2007] Schneier, B. (2007). Information Security and Externalities. *European Network and Information Security Agency Quarterly*, 2(4):3–4.

[Scholl, 2001] Scholl, H. J. (2001). Applying Stakeholder Theory to E-government: Benefits and Limits. In Schmid, B., Stanoevska-Slabeva, K., and Tschammer, V., editors, *Towards the E-Society: E-Commerce, E-Business, and E-Government*, chapter 54, pages 735–747. Springer.

[Schonlau et al., 2002] Schonlau, M., Fricker, Jr., R. D., and Elliott, M. N. (2002). *Conducting Research Surveys via E-mail and the Web*. RAND Corporation.

[Scott et al., 2004] Scott, E., Zadirov, A., Feinberg, S., and Jayakody, R. (2004). The Alignment of Software Testing Skills of IS Students with Industry Practices – A South African Perspective. *Journal of Information Technology Education*, 3:161–172.

[Scott and Orlikowski, 2009] Scott, S. V. and Orlikowski, W. J. (2009). "Getting the Truth": Exploring the Material Grounds of Institutional Dynamics in Social Media. In *EGOS '09: Proceedings of the 25th European Group for Organizational Studies Colloquium*.

[Seaman, 1999] Seaman, C. B. (1999). Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572.

[Sedano et al., 2017] Sedano, T., Ralph, P., and Péraire, C. (2017). Software Development Waste. In *ICSE '17: Proceedings of the 39th International Conference on Software Engineering*, pages 130–140. IEEE Computer Society.

[Shachaf and Rosenbaum, 2009] Shachaf, P. and Rosenbaum, H. (2009). Online Social Reference: A Research Agenda Through a STIN Framework. In *iConference '09: Proceedings of the 4th iConference*.

[Shafique and Labiche, 2015] Shafique, M. and Labiche, Y. (2015). A systematic review of state-based test tools. *International Journal on Software Tools for Technology Transfer*, 17(1):59–76.

[Shah and Harrold, 2010] Shah, H. and Harrold, M. J. (2010). Case Study: Studying Human and Social Aspects of Testing in a Service-Based Software Company. In *CHASE '10: Proceedings of the 3rd International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 102–108. ACM Press.

[Shah, 2004] Shah, S. (2004). The researcher/interviewer in intercultural context: a social intruder! *British Educational Research Journal*, 30(4):549–575.

[Shahin et al., 2017] Shahin, M., Babar, M. A., Zahedi, M., and Zhu, L. (2017). Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges. In *ESEM '17: Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 111–120. IEEE Computer Society.

[Sharp et al., 2009] Sharp, H., Baddoo, N., Beecham, S., Hall, T., and Robinson, H. (2009). Models of motivation in software engineering. *Information and Software Technology*, 51(1):219–233.

[Sharp et al., 1999] Sharp, H., Finkelstein, A., and Galal, G. (1999). Stakeholder Identification in the Requirements Engineering Process. In *DEXA '99: Proceedings of the Tenth International Workshop on Database and Expert Systems Applications*, pages 387–391. IEEE Computer Society.

[Sharp and Hall, 2009] Sharp, H. and Hall, T. (2009). An initial investigation of software practitioners' motivation. In *CHASE '09: Proceedings of the 2nd International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 84–91. IEEE Computer Society.

[Shaye, 2008] Shaye, S. D. (2008). Transitioning a Team to Agile Test Methods. In *Agile '08: Proceedings of the 2008 Agile Conference*, pages 470–477. IEEE Computer Society.

[Sheard, 2001] Sheard, S. A. (2001). Evolution of the Frameworks Quagmire. *IEEE Computer*, 34(7):96–98.

[Shepard and Kelly, 2001] Shepard, T. and Kelly, D. (2001). How to do Inspections When There is No Time. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 718–719. IEEE Computer Society.

[Shoemaker et al., 2002] Shoemaker, P. J., Eichholz, M., and Skewes, E. A. (2002). Item Nonresponse: Distinguishing Between Don't Know and Refuse. *International Journal of Public Opinion Research*, 14(2):193–201.

[Shull et al., 2002] Shull, F., Basili, V., Boehm, B. W., Brown, A. W., Costa, P., Lindvall, M., Port, D., Rus, I., Tesoriero, R., and Zelkowitz, M. (2002). What We Have Learned About Fighting Defects. In *METRICS '02: Proceedings of the 8th International Symposium on Software Metrics*, pages 249–258. IEEE Computer Society.

[Shull et al., 2012] Shull, F., Feldmann, R. L., Seaman, C., Regardie, M., and Godfrey, S. (2012). Fully Employing Software Inspections Data. *Innovations in Systems and Software Engineering*, 8(4):243–254.

[Shull et al., 2008] Shull, F. J., Carver, J. C., Vegas, S., and Juristo, N. (2008). The role of replications in Empirical Software Engineering. *Empirical Software Engineering*, 13(2):211–218.

[Sillitti, 2009] Sillitti, A. (2009). Designing Empirical Studies: Assessing the Effectiveness of Agile Methods. *ACM SIGSOFT Software Engineering Notes*, 34(5):35–37.

[Sim et al., 2001] Sim, S. E., Singer, J., and Storey, M.-A. (2001). Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research. *Empirical Software Engineering*, 6(1):85–93.

[Singer et al., 2008] Singer, J., Sim, S. E., and Lethbridge, T. C. (2008). Software Engineering Data Collection for Field Studies. In Shull, F., Singer, J., and Sjøberg, D. I. K., editors, *Guide to Advanced Empirical Software Engineering*, pages 9–34. Springer.

[Singer and Vinson, 2002] Singer, J. and Vinson, N. G. (2002). Ethical Issues in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, 28(12):1171–1180.

[Singh et al., 2009] Singh, A., Taneja, A., and Mangalaraj, G. (2009). Creating Online Surveys: Some Wisdom from the Trenches Tutorial. *IEEE Transactions on Professional Communication*, 52(2):197–212.

[Singleton et al., 2011] Singleton, R., Toombs, L. A., Taneja, S., Larkin, C., and Pryor, M. G. (2011). Workplace conflict: A strategic leadership imperative. *International Journal of Business and Public Administration*, 8(1):149–163.

[Sinkovics et al., 2013] Sinkovics, N., Sinkovics, R. R., and Jean, R.-J. B. (2013). The internet as an alternative path to internationalization? *International Marketing Review*, 30(2):130–155.

[Sinnhofer et al., 2015] Sinnhofer, A. D., Raschke, W., Steger, C., and Kreiner, C. (2015). Evaluation paradigm selection according to Common Criteria for an incremental product development. In *MILS '15: Proceedings of the 1st International Workshop on MILS: Architecture and Assurance for Secure Systems*.

[Siponen et al., 2014] Siponen, M., Mahmood, M. A., and Pahnila, S. (2014). Employees' adherence to information security policies: An exploratory field study. *Information & Management*, 51(2):217–224.

[Sitnikova et al., 2007] Sitnikova, E., Kroeger, T., and Cook, S. (2007). Software and Systems Engineering Process Capability in the South Australian Defence Industry. *Innovations in Systems and Software Engineering*, 3(2):129–139.

[Sjøberg and Grimstad, 2010] Sjøberg, D. I. K. and Grimstad, S. (2010). Software Engineering - Why, What, How and What's Next. In Tveito, A., Bruaset, A. M., and Lysne, O., editors, *Simula Research Laboratory*, pages 363–368. Springer.

[Sjøberg et al., 2005] Sjøberg, D. I. K., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanović, A., Liborg, N.-K., and Rekdal, A. C. (2005). A Survey of Controlled Experiments in Software Engineering. *IEEE Transactions on Software Engineering*, 31(9):733–753.

[Slyngstad et al., 2008] Slyngstad, O. P. N., Li, J., Conradi, R., and Babar, M. A. (2008). Identifying and Understanding Architectural Risks in Software Evolution: An Empirical Study. In Jedlitschka, A. and Salo, O., editors, *PROFES '08: Proceedings of the 9th International Conference on Product-Focused Software Process Improvement*, pages 400–414. Springer.

[Smelter and Moseley, 2018] Smelter, A. and Moseley, H. N. B. (2018). A Python library for FAIRer access and deposition to the Metabolomics Workbench Data Repository. *Metabolomics*, 14(5).

[Smith et al., 2013] Smith, E., Loftin, R., Murphy-Hill, E., Bird, C., and Zimmermann, T. (2013). Improving Developer Participation Rates in Surveys. In *CHASE '13:*

Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering, pages 89–92. IEEE Computer Society.

[Smith and Hasnas, 1999] Smith, H. J. and Hasnas, J. (1999). Ethics and Information Systems: The Corporate Domain. *MIS Quarterly*, 23(1):109–127.

[Smith, 2007] Smith, R. E. (2007). Trends in Security Product Evaluations. *Information Systems Security*, 16(4):203–216.

[Smith, 2015] Smith, R. E. (2015). *Elementary Information Security*. Jones & Bartlett Learning, second edition.

[Smuts et al., 2010] Smuts, H., van der Merwe, A., Kotzé, P., and Loock, M. (2010). Critical Success Factors for Information Systems Outsourcing Management: A Software Development Lifecycle View. In *SAICSIT '10: Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 304–313. ACM Press.

[Springer, 2020] Springer (2020). Springer Link. `https://link.springer.com/`. Retrieved: 16th of January, 2020.

[Staab, 2002] Staab, T. C. (2002). Using SW-TMM to Improve the Testing Process. *CrossTalk: The Journal of Defense Software Engineering*, 15(11):13–16.

[Stallings, 2017] Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice*. Pearson Education, seventh edition.

[Stamp, 2011] Stamp, M. (2011). *Information Security: Principles and Practice*. John Wiley & Sons, second edition.

[Stanton, 2007] Stanton, J. M. (2007). Empirical vs. Non-Empirical Work in Information Systems Security: A Review and Analysis of Published Articles 1995-2005. In *HAISA '07: Proceedings of the International Symposium on Human Aspects of Information Security & Assurance*, pages 141–155. University of Plymouth.

[Staples et al., 2007] Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., and Murphy, R. (2007). An exploratory study of why organizations do not adopt CMMI. *Journal of Systems and Software*, 80(6):883–895.

[Stober and Hansmann, 2010] Stober, T. and Hansmann, U. (2010). *Agile Software Development: Best Practices for Large Software Development Projects*. Springer.

[Stokes, 2001] Stokes, A. (2001). Using telementoring to deliver training to SMEs: a pilot study. *Education + Training*, 43(6):317–324.

[Storey et al., 2008] Storey, M.-A., Ryall, J., Bull, R. I., Myers, D., and Singer, J. (2008). TODO or To Bug: Exploring How Task Annotations Play a Role in the Work Practices of Software Developers. In *ICSE '08: Proceedings of the 30th International Conference on Software Engineering*, pages 251–260. ACM Press.

[Street and Meister, 2004] Street, C. T. and Meister, D. B. (2004). Small Business Growth and Internal Transparency: The Role of Information Systems. *MIS Quarterly*, 28(3):473–506.

[Stringfellow and York, 2004] Stringfellow, C. V. and York, D. L. (2004). An Example of Practical Component Testing. *Journal of Computing Sciences in Colleges*, 19(4):203–210.

[Su et al., 2017] Su, T., Wu, K., Miao, W., Pu, G., He, J., Chen, Y., and Su, Z. (2017). A Survey on Data-Flow Testing. *ACM Computing Surveys*, 50(1):5:1–5:35.

[Sumrell, 2007] Sumrell, M. (2007). From Waterfall to Agile - How does a QA Team Transition? In *AGILE '07: Proceedings of the AGILE 2007 Conference*, pages 291–295. IEEE Computer Society.

[Sung and Paynter, 2006] Sung, P. W.-B. and Paynter, J. (2006). Software Testing Practices in New Zealand. In *NACCQ '06: Proceedings of the 19th Annual Conference of the National Advisory Committee on Computing Qualifications*, pages 273–282.

[Suri, 2013] Suri, V. R. (2013). *ICT Use in Knowledge Work: GIS and Historiographical Practices*. PhD thesis, Indiana University.

[Sutherland et al., 2007] Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. (2007). Distributed Scrum: Agile Project Management with Outsourced Development Teams. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society.

[Swan, 2008] Swan, A. (2008). Study on the Availability of UK Academic "Grey Literature" to UK SMEs. `https://www.webarchive.org.uk/wayback/archive/20140613220103/http://www.jisc.ac.uk/media/documents/aboutus/workinggroups/greyliteraturereport.pdf`. Retrieved: 16th of January, 2020.

[Swinkels, 2000] Swinkels, R. (2000). A comparison of TMM and other Test Process Improvement Models. Technical report, Frits Philips Institute 12-4-1-FP.

[Tahir and Ahmad, 2010] Tahir, A. and Ahmad, R. (2010). Requirement Engineering Practices - An Empirical Study. In *CiSE '10: Proceedings of the 2010 International Conference on Computational Intelligence and Software Engineering*. IEEE Computer Society.

[Tahir et al., 2015] Tahir, A., MacDonell, S., and Buchan, J. (2015). A Study of the Relationship Between Class Testability and Runtime Properties. In Maciaszek, L. A. and Filipe, J., editors, *Evaluation of Novel Approaches to Software Engineering*, volume 551 of *Communications in Computer and Information Science*, pages 63–78. Springer.

[Taipale et al., 2005] Taipale, O., Smolander, K., and Kälviäinen, H. (2005). Finding and Ranking Research Directions for Software Testing. In *EuroSPI '05: Proceedings of the 12th European Conference on Software Process Improvement*, pages 39–48. Springer.

[Takanen et al., 2008] Takanen, A., DeMott, J. D., and Miller, C. (2008). *Fuzzing for Software Security Testing and Quality Assurance*. Artech House.

[Talby et al., 2006] Talby, D., Keren, A., Hazzan, O., and Dubinsky, Y. (2006). Agile Software Testing in a Large-Scale Project. *IEEE Software*, 23(4):30–37.

[Talib et al., 2010] Talib, S., Clarke, N. L., and Furnell, S. M. (2010). An Analysis of Information Security Awareness within Home and Work Environments. In *ARES '10: Proceedings of the 5th International Conference on Availability, Reliability and Security*, pages 196–203. IEEE Computer Society.

[Tanner, 2003] Tanner, F. R. (2003). On Motivating Engineers. In *IEMC '03: Proceedings of the 2003 IEEE International Engineering Management Conference*, pages 214–218. IEEE Computer Society.

[Tashakkori, 2009] Tashakkori, A. (2009). Are We There Yet?: The State of the Mixed Methods Community. *Journal of Mixed Methods Research*, 3(4):287–291.

[Tashakkori and Teddlie, 2010] Tashakkori, A. and Teddlie, C. (2010). Putting the Human Back in "Human Research Methodology": The Researcher in Mixed Methods Research. *Journal of Mixed Methods Research*, 4(4):271–277.

[Tatnall, 2005] Tatnall, A. (2005). Actor-Network Theory in Information Systems Research. In Khosrow-Pour, M., editor, *Encyclopedia of Information Science and Technology*, pages 42–46. Idea Group Reference.

[Tatnall and Gilding, 1999] Tatnall, A. and Gilding, A. (1999). Actor-Network Theory and Information Systems Research. In *ACIS '99: Proceedings of the 10th Australasian Conference on Information Systems*, pages 955–966. Victoria University of Wellington.

[Taylor-Smith, 2016] Taylor-Smith, E. (2016). *Participation Space Studies: a socio-technical exploration of activist and community groups' use of online and offline spaces to support their work*. PhD thesis, Edinburgh Napier University.

[Taylor-Smith and Smith, 2018] Taylor-Smith, E. and Smith, C. F. (2018). Investigating the online and offline contexts of day-to-day democracy as participation spaces. *Information, Communication & Society*, pages 1–18.

[Teddlie and Tashakkori, 2012] Teddlie, C. and Tashakkori, A. (2012). Common "Core" Characteristics of Mixed Methods Research: A Review of Critical Issues and Call for Greater Convergence. *American Behavioral Scientist*, 56(6):774–788.

[Teh et al., 2013] Teh, N., Schuff, D., Johnson, S., and Geddes, D. (2013). Can Work Be Fun? Improving Task Motivation and Help-Seeking Through Game Mechanics. In *ICIS '13: Proceedings of the 34th International Conference on Information Systems*.

[Telang and Wattal, 2007] Telang, R. and Wattal, S. (2007). An Empirical Analysis of the Impact of Software Vulnerability Announcements on Firm Stock Price. *IEEE Transactions on Software Engineering*, 33(8):544–557.

[Teruel-Carrizosa, 2010] Teruel-Carrizosa, M. (2010). Gibrat's Law and the Learning Process. *Small Business Economics*, 34(4):355–373.

[Tervonen and Harjumaa, 2004] Tervonen, I. and Harjumaa, L. (2004). Encouraging SME Software Companies to Adopt Software Reviews in Their Entirety. In *PNSQC '04: Proceedings of the 22nd Annual Pacific Northwest Software Quality Conference*, pages 525–536.

[Testa, 2009] Testa, L. (2009). *Growing Software: Proven Strategies for Managing Software Engineers*. No Starch Press.

[The Economist, 2003] The Economist (2003). Building a better bug-trap. `https://www.economist.com/technology-quarterly/2003/06/21/building-a-better-bug-trap`. Retrieved: 16th of January, 2020.

[The MITRE Corporation, 2020] The MITRE Corporation (2020). Common Weakness Enumeration: A Community-Developed List of Software Weakness Types. `http://cwe.mitre.org/`. Retrieved: 16th of January, 2020.

[Theocharis et al., 2015] Theocharis, G., Kuhrmann, M., Münch, J., and Diebold, P. (2015). Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices. In Abrahamsson, P., Corral, L., Oivo, M., and Russo, B., editors, *PROFES '15: Proceedings of the 16th International Conference on Product-Focused Software Process Improvement*, pages 149–166. Springer.

[Thompson, 2003] Thompson, H. H. (2003). Why Security Testing Is Hard. *IEEE Security & Privacy*, 1(4):83–86.

[Thompson, 2005] Thompson, H. H. (2005). Application Penetration Testing. *IEEE Security & Privacy*, 3(1):66–69.

[Thompson et al., 2002] Thompson, H. H., Whittaker, J. A., and Mottay, F. E. (2002). Software Security Vulnerability Testing in Hostile Environments. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 260–264. ACM Press.

[Thompson and Edwards, 2008] Thompson, J. B. and Edwards, H. M. (2008). Advancing Industry-Related Elements: The Meat on the Curricula Bones. In *COMPSAC '08: Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 447–454. IEEE Computer Society.

[Thomson and von Solms, 2006] Thomson, K.-L. and von Solms, R. (2006). Towards an Information Security Competence Maturity Model. *Computer Fraud & Security*, 2006(5):11–15.

[Thomson et al., 2006] Thomson, K.-L., von Solms, R., and Louw, L. (2006). Cultivating an organizational information security culture. *Computer Fraud & Security*, 2006(10):7–11.

[Thörn, 2010] Thörn, C. (2010). Current state and potential of variability management practices in software-intensive SMEs: Results from a regional industrial survey. *Information and Software Technology*, 52(4):411–421.

[Thörn and Gustafsson, 2008] Thörn, C. and Gustafsson, T. (2008). Uptake of Modeling Practices in SMEs: Initial Results from an Industrial Survey. In *MiSE '08: Proceedings of the 2008 International Workshop on Models in Software Engineering*, pages 21–26. ACM Press.

[Thorne, 2000] Thorne, S. (2000). Data analysis in qualitative research. *Evidence-Based Nursing*, 3(3):68–70.

[Tian, 2005] Tian, J. (2005). *Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement*. John Wiley & Sons.

[Tichy and Padberg, 2007] Tichy, W. F. and Padberg, F. (2007). Empirical Methods in Software Engineering Research. In *ICSE-C '07: Proceedings of the 29th International Conference on Software Engineering Companion*, pages 163–164. IEEE Computer Society.

[Tillmann et al., 2010] Tillmann, N., de Halleux, J., and Xie, T. (2010). Parameterized Unit Testing: Theory and Practice. In *ICSE '10: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, pages 483–484. ACM Press.

[TMMi Foundation, 2018] TMMi Foundation (2018). Test Maturity Model integration (TMMi) - Release 1.2. `https://tmmi.org/tm6/wp-content/uploads/2018/11/TMMi-Framework-R1-2.pdf`. Retrieved: 16th of January, 2020.

[Tödtling and Kaufmann, 2001] Tödtling, F. and Kaufmann, A. (2001). The Role of the Region for Innovation Activities of SMEs. *European Urban and Regional Studies*, 8(3):203–215.

[Tomayko and Hazzan, 2004] Tomayko, J. E. and Hazzan, O. (2004). *Human Aspects of Software Development*. Charles River Media.

[Torkar and Mankefors, 2003] Torkar, R. and Mankefors, S. (2003). A Survey on Testing and Reuse. In *Proceedings of the IEEE International Conference on Software-Science, Technology and Engineering*, pages 164–173. IEEE Computer Society.

[Tosun et al., 2009] Tosun, A., Bener, A., and Turhan, B. (2009). Implementation of a Software Quality Improvement Project in an SME: A Before and After Comparison. In *SEAA '09: Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications*, pages 203–209. IEEE Computer Society.

[Tourangeau et al., 2000] Tourangeau, R., Rips, L. J., and Rasinski, K. (2000). *The Psychology of Survey Response*. Cambridge University Press.

[Tourangeau and Yan, 2007] Tourangeau, R. and Yan, T. (2007). Sensitive Questions in Surveys. *Psychological Bulletin*, 133(5):859–883.

[Trew, 2007] Trew, T. (2007). Chasing Rainbows: Improving Software Testing in the Real World. In *ISSTA '07: Proceedings of the 2007 International Symposium on Software Testing and Analysis*, pages 95–96. ACM Press.

[Trist and Bamforth, 1951] Trist, E. L. and Bamforth, K. W. (1951). Some Social and Psychological Consequences of the Longwall Method of Coal-Getting: An Examination of the Psychological Situation and Defences of a Work Group in Relation to the Social Structure and Technological Content of the Work System. *Human Relations*, 4(1):3–38.

[TUAC, 2001] TUAC (2001). TUAC Comment on Job Tenure. `https://old.tuac.org/News/njobtenure.htm`. Retrieved: 16th of January, 2020.

[Turner, 2005] Turner, F. (2005). Actor-Networking the News. *Social Epistemology: A Journal of Knowledge, Culture and Policy*, 19(4):321–324.

[Türpe, 2008] Türpe, S. (2008). Security Testing: Turning Practice into Theory. In *ICSTW '08: Proceedings of the 2008 IEEE International Conference on Software Testing Verification and Validation Workshop*, pages 294–302. IEEE Computer Society.

[Uffen and Breitner, 2015] Uffen, J. and Breitner, M. H. (2015). Management of Technical Security Measures: An Empirical Examination of Personality Traits and Behavioral Intentions. In *Standards and Standardization: Concepts, Methodologies, Tools, and Applications*, volume 2, pages 836–853. IGI Global.

[Umbach, 2004] Umbach, P. D. (2004). Web Surveys: Best Practices. *New Directions for Institutional Research*, 2004(121):23–38.

[Urquhart and Currell, 2010] Urquhart, C. and Currell, R. (2010). Home uterine monitoring: A case of telemedicine failure? *Health Informatics Journal*, 16(3):165–175.

[Urquhart and Currell, 2016] Urquhart, C. and Currell, R. (2016). Systematic Reviews and Meta-Analysis of Health IT. In Ammenwerth, E. and Rigby, M., editors, *Evidence-Based Health Informatics: Promoting Safety and Efficiency through Scientific Methods and Ethical Policy*, pages 262–274. IOS Press.

[Uwano et al., 2008] Uwano, H., Monden, A., and Matsumoto, K. (2008). Are Good Code Reviewers Also Good at Design Review? In *ESEM '08: Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement*, pages 351–353. ACM Press.

[Valtanen and Ahonen, 2008] Valtanen, A. and Ahonen, J. J. (2008). Big Improvements with Small Changes: Improving the Processes of a Small Software Company.

In Jedlitschka, A. and Salo, O., editors, *PROFES '08: Proceedings of the 9th International Conference on Product-Focused Software Process Improvement*, pages 258–272. Springer.

[van der Merwe, 2010] van der Merwe, R. (2010). Investigating direct deliberative governance in online social media. Technical report, The Open University.

[van Genuchten and Hatton, 2013] van Genuchten, M. and Hatton, L. (2013). Quantifying Software's Impact. *IEEE Computer*, 46(10):66–72.

[van Selm and Jankowski, 2006] van Selm, M. and Jankowski, N. W. (2006). Conducting Online Surveys. *Quality and Quantity*, 40(3):435–456.

[van Wyk, 2013] van Wyk, K. (2013). Adapting Penetration Testing for Software Development Purposes. `https://www.us-cert.gov/bsi/articles/best-practices/security-testing/adapting-penetration-testing-software-development-purposes`. Retrieved: 16th of January, 2020.

[Venkatesh et al., 2013] Venkatesh, V., Brown, S. A., and Bala, H. (2013). Bridging the Qualitative-Quantitative Divide: Guidelines for Conducting Mixed Methods Research in Information Systems. *MIS Quarterly*, 37(1):21–54.

[Venkatesh et al., 2016] Venkatesh, V., Brown, S. A., and Sullivan, Y. W. (2016). Guidelines for Conducting Mixed-methods Research: An Extension and Illustration. *Journal of the Association for Information Systems*, 17(7):435–495.

[Vermaas et al., 2011] Vermaas, P., Kroes, P., van de Poel, I., Franssen, M., and Houkes, W. (2011). *A Philosophy of Technology: From Technical Artefacts to Sociotechnical Systems*. Morgan & Claypool.

[Vieira et al., 2006] Vieira, F. E., Martins, F., Silva, R., Menezes, R., and Braga, M. (2006). On the Idea of Using Nature-Inspired Metaphors to Improve Software Testing. In Maglogiannis, I., Karpouzis, K., and Bramer, M., editors, *Artificial Intelligence Applications and Innovations*, volume 204 of *IFIP International Federation for Information Processing*, pages 541–548. Springer.

[Vijayasarathy and Butler, 2016] Vijayasarathy, L. R. and Butler, C. W. (2016). Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? *IEEE Software*, 33(5):86–94.

[Villar-Onrubia and Rajpal, 2016] Villar-Onrubia, D. and Rajpal, B. (2016). Online international learning: Internationalising the curriculum through virtual mobility at Coventry University. *Perspectives: Policy and Practice in Higher Education*, 20(2-3):75–82.

[Villela et al., 2014] Villela, K., Silva, A., Vale, T., and de Almeida, E. S. (2014). A Survey on Software Variability Management Approaches. In *SPLC '14: Proceedings of the 18th International Software Product Line Conference - Volume 1*, pages 147–156. ACM Press.

[Voas and Miller, 2012] Voas, J. and Miller, K. W. (2012). Software Testing: What Goes Around Comes Around. *IT Professional*, 14(3):4–5.

[Vogel, 2011] Vogel, D. A. (2011). *Medical Device Software Verification, Validation and Compliance*. Artech House.

[von Solms, 2001a] von Solms, B. (2001a). Corporate Governance and Information Security. *Computers & Security*, 20(3):215–218.

[von Solms, 2001b] von Solms, B. (2001b). Information Security - A Multidimensional Discipline. *Computers & Security*, 20(6):504–508.

[von Solms and von Solms, 2004] von Solms, B. and von Solms, R. (2004). The 10 deadly sins of information security management. *Computers & Security*, 23(5):371–376.

[von Solms and von Solms, 2009] von Solms, S. H. and von Solms, R. (2009). *Information Security Governance*. Springer.

[Votta, Jr., 1993] Votta, Jr., L. G. (1993). Does Every Inspection Need a Meeting? *ACM SIGSOFT Software Engineering Notes*, 18(5):107–114.

[Vrhovec et al., 2015] Vrhovec, S. L. R., Hovelja, T., Vavpotič, D., and Krisper, M. (2015). Diagnosing organizational risks in software projects: Stakeholder resistance. *International Journal of Project Management*, 33(6):1262–1273.

[Walker, 2008] Walker, S. (2008). Digital design in social action settings: A review through a sociotechnical lens. In *CIRN '08: Proceedings of the 5th Prato Community Informatics & Development Informatics Conference 2008: ICTs for Social Inclusion: What is the Reality?* Centre for Community Networking Research.

[Walker and Creanor, 2009] Walker, S. and Creanor, L. (2009). The STIN in the Tale: A Socio-technical Interaction Perspective on Networked Learning. *Educational Technology & Society*, 12(4):305–316.

[Walsham, 1997] Walsham, G. (1997). Actor-Network Theory and IS Research: Current Status and Future Prospects. In Lee, A. S., Liebenau, J., and DeGross, J. I., editors, *Information Systems and Qualitative Research*, pages 466–480. Springer.

[Wang and Guo, 2009] Wang, J. A. and Guo, M. (2009). Security Data Mining in an Ontology for Vulnerability Management. In *IJCBS 09: Proceedings of the 2009 International Joint Conferences on Bioinformatics, Systems Biology and Intelligent Computing*, pages 597–603. IEEE Computer Society.

[Ward et al., 2001] Ward, R. P., Fayad, M. E., and Laitinen, M. (2001). Software Process Improvement in the Small. *Communications of the ACM*, 44(4):105–107.

[Watkins, 2009] Watkins, J. (2009). *Agile Testing: How to Succeed in an Extreme Testing Environment*. Cambridge University Press.

[Watkins and Mills, 2011] Watkins, J. and Mills, S. (2011). *Testing IT: An Off-the-Shelf Software Testing Process*. Cambridge University Press, second edition.

[Watson et al., 2009] Watson, P. G., Duquenoy, P., Brennan, M., Jones, M., and Walkerdine, J. (2009). Towards an Ethical Interaction Design: the issue of including stakeholders in law-enforcement software development. In *OzCHI '09: Proceedings of the 21st Australian Conference on Computer-Human Interaction*, pages 313–316. ACM Press.

[Wäyrynen et al., 2004] Wäyrynen, J., Bodén, M., and Boström, G. (2004). Security Engineering and eXtreme Programming: An Impossible Marriage? In Zannier, C., Erdogmus, H., and Lindstrom, L., editors, *Extreme Programming and Agile Methods - XP/Agile Universe 2004*, pages 117–128. Springer.

[Weber, 2004] Weber, R. (2004). The Rhetoric of Positivism Versus Interpretivism: A Personal View. *MIS Quarterly*, 28(1):iii–xii.

[Weißenfels et al., 2016] Weißenfels, S., Ebner, K., Dittes, S., and Smolnik, S. (2016). Does the IS Artifact Matter in Sociomateriality Research? A Literature Review of Empirical Studies. In *HICSS '16: Proceedings of the 49th Annual Hawaii International Conference on System Sciences*, pages 1997–2006. IEEE Computer Society.

[West, 2011] West, D. (2011). Water-Scrum-Fall Is The Reality Of Agile For Most Organizations Today. Technical report, Forrester.

[Weyuker, 2011] Weyuker, E. J. (2011). Empirical Software Engineering Research - The Good, The Bad, The Ugly. In *ESEM '11: Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement*, pages 1–9. IEEE Computer Society.

[Whitehead et al., 2010] Whitehead, J., Mistrík, I., Grundy, J., and van der Hoek, A. (2010). Collaborative Software Engineering: Concepts and Techniques. In Mistrík, I., Grundy, J., van der Hoek, A., and Whitehead, J., editors, *Collaborative Software Engineering*. Springer.

[Whittaker, 2000] Whittaker, J. A. (2000). What Is Software Testing? And Why Is It So Hard? *IEEE Software*, 17(1):70–79.

[Whittaker, 2012] Whittaker, J. A. (2012). The 10-Minute Test Plan. *IEEE Software*, 29(6):70–77.

[Whittaker and Thompson, 2004] Whittaker, J. A. and Thompson, H. H. (2004). *How to Break Software Security: Effective Techniques for Security Testing*. Addison-Wesley.

[Whitworth, 2008] Whitworth, E. (2008). Experience Report: The Social Nature of Agile Teams. In *AGILE '08: Proceedings of the AGILE 2008 Conference*, pages 429–435. IEEE Computer Society.

[Whitworth and Biddle, 2007] Whitworth, E. and Biddle, R. (2007). The Social Nature of Agile Teams. In *AGILE '07: Proceedings of the AGILE 2007 Conference*, pages 26–36. IEEE Computer Society.

[Wicks, 2005] Wicks, M. (2005). A Software Engineering Survey. Technical report, Heriot-Watt University.

[Wiederseiner et al., 2010] Wiederseiner, C., Jolly, S., Garousi, V., and Eskandar, M. (2010). An Open-Source Tool for Automated Generation of Black-Box xUnit Test Code and Its Industrial Evaluation. In Bottaci, L. and Fraser, G., editors, *Testing - Practice and Research Techniques*, volume 6303 of *Lecture Notes in Computer Science*, pages 118–128. Springer.

[Wiegers, 1996] Wiegers, K. E. (1996). Software Process Improvement: Ten Traps to Avoid. *Software Development*, pages 51–58.

[Wielki, 2011] Wielki, J. (2011). The usefulness of the stakeholder theory in an analysis of the internet-impacted business environment of contemporary organizations. In

ETHICOMP '11: Proceedings of the Twelfth International Conference on the Social and Ethical Impacts of Information and Communication Technology, pages 494–502. Sheffield Hallam University.

[Wildy and Clarke, 2009] Wildy, H. and Clarke, S. (2009). Using cognitive interviews to pilot an international survey of principal preparation: A Western Australian perspective. *Educational Assessment, Evaluation and Accountability*, 21(2):105–117.

[Williams, 2009] Williams, P. A. H. (2009). What Does Security Culture Look Like For Small Organizations? In *Proceedings of the 7th Australian Information Security Management Conference*, pages 48–54. Edith Cowan University.

[Winner, 1993] Winner, L. (1993). Upon Opening the Black Box and Finding It Empty: Social Constructivism and the Philosophy of Technology. *Science, Technology, & Human Values*, 18(3):362–378.

[Witschey et al., 2013] Witschey, J., Murphy-Hill, E., and Xiao, S. (2013). Conducting Interview Studies: Challenges, Lessons Learned, and Open Questions. In *CESI'13: Proceedings of the 1st International Workshop on Conducting Empirical Studies in Industry*, pages 51–54. IEEE Computer Society.

[Wohlin, 2013] Wohlin, C. (2013). Empirical Software Engineering Research with Industry: Top 10 Challenges. In *CESI'13: Proceedings of the 1st International Workshop on Conducting Empirical Studies in Industry*, pages 43–46. IEEE Computer Society.

[Wohlin, 2014] Wohlin, C. (2014). Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In *EASE '14: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, pages 38:1–38:10. ACM Press.

[Wohlin and Aurum, 2015] Wohlin, C. and Aurum, A. (2015). Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, 20(6):1427–1455.

[Wohlin et al., 2003] Wohlin, C., Höst, M., and Henningsson, K. (2003). Empirical Research Methods in Software Engineering. In Conradi, R. and Wang, A. I., editors, *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, volume 2765, pages 7–23. Springer.

[Wojcicki and Strooper, 2006] Wojcicki, M. A. and Strooper, P. (2006). A State-of-Practice Questionnaire on Verification and Validation for Concurrent Programs. In

PADTAD '06: Proceedings of the 2006 workshop on Parallel and Distributed Systems: Testing and Debugging, pages 1–10. ACM Press.

[Wong and Aspinwall, 2004] Wong, K. Y. and Aspinwall, E. (2004). Characterizing knowledge management in the small business environment. *Journal of Knowledge Management*, 8(3):44–61.

[Wong and Aspinwall, 2005] Wong, K. Y. and Aspinwall, E. (2005). An empirical study of the important factors for knowledge-management adoption in the SME sector. *Journal of Knowledge Management*, 9(3):64–82.

[Wood, 1997] Wood, C. C. (1997). Policies Alone Do Not Constitute a Sufficient Awareness Effort. *Computer Fraud & Security*, 1997(12):14–19.

[Wood et al., 1999] Wood, M., Daly, J., Miller, J., and Roper, M. (1999). Multi-method research: An empirical investigation of object-oriented technology. *Journal of Systems and Software*, 48(1):13–26.

[Wood et al., 1997] Wood, M., Roper, M., Brooks, A., and Miller, J. (1997). Comparing and Combining Software Defect Detection Techniques: A Replicated Empirical Study. *ACM SIGSOFT Software Engineering Notes*, 22(6):262–277.

[Woschke et al., 2017] Woschke, T., Haase, H., and Kratzer, J. (2017). Resource scarcity in SMEs: effects on incremental and radical innovations. *Management Research Review*, 40(2):195–217.

[Würsch et al., 2013] Würsch, M., Giger, E., and Gall, H. C. (2013). Evaluating a Query Framework for Software Evolution Data. *ACM Transactions on Software Engineering and Methodology*, 22(4):38:1–38:38.

[Wurster and van Oorschot, 2008] Wurster, G. and van Oorschot, P. C. (2008). The Developer is the Enemy. In *NSPW '08: Proceedings of the 2008 New Security Paradigms Workshop*, pages 89–97. ACM Press.

[Wysopal et al., 2006] Wysopal, C., Nelson, L., Zovi, D. D., and Dustin, E. (2006). *The Art of Software Security Testing: Identifying Software Security Flaws*. Addison-Wesley.

[Xiao et al., 2007] Xiao, M., El-Attar, M., Reformat, M., and Miller, J. (2007). Empirical evaluation of optimization algorithms when used in goal-oriented automated test data generation techniques. *Empirical Software Engineering*, 12(2):183–239.

[Xiong, 2011] Xiong, J. (2011). *New Software Engineering Paradigm Based on Complexity Science*. Springer.

[Yang et al., 2016] Yang, R., Li, G., Lau, W. C., Zhang, K., and Hu, P. (2016). Model-based Security Testing: An Empirical Study on OAuth 2.0 Implementations. In *ASIACCS '16: Proceedings of the 11th ACM Asia Conference on Computer and Communications Security*, pages 651–662. ACM Press.

[Yeo, 2013] Yeo, J. (2013). Using penetration testing to enhance your company's security. *Computer Fraud & Security*, 2013(4):17–20.

[Yilmaz and Phillips, 2006] Yilmaz, L. and Phillips, J. (2006). Organization-Theoretic Perspective for Simulation Modeling of Agile Software Processes. In Wang, Q., Pfahl, D., Raffo, D. M., and Wernick, P., editors, *Software Process Change*, volume 3966, pages 234–241. Springer.

[Zabicki and Ellis, 2017] Zabicki, R. and Ellis, S. R. (2017). Penetration Testing. In Vacca, J. R., editor, *Computer and Information Security Handbook*, chapter 75, pages 1031–1038. Elsevier, third edition.

[Zangooei et al., 2012] Zangooei, T., Mansoori, M., and Welch, I. (2012). A Hybrid Recognition and Recall Based Approach in Graphical Passwords. In *OzCHI '12: Proceedings of the 24th Australian Conference on Computer-Human Interaction*, pages 665–673. ACM Press.

[Zelkowitz and Wallace, 1998] Zelkowitz, M. V. and Wallace, D. R. (1998). Experimental Models for Validating Technology. *IEEE Computer*, 31(5):23–31.

[Zelkowitz et al., 1984] Zelkowitz, M. V., Yeh, R. T., Hamlet, R. G., Gannon, J. D., and Basili, V. R. (1984). Software Engineering Practices in the US and Japan. *IEEE Computer*, 17(6):57–66.

[Zhang et al., 2014] Zhang, X., Stafford, T. F., Dhaliwal, J. S., Gillenson, M. L., and Moeller, G. (2014). Sources of conflict between developers and testers in software development. *Information & Management*, 51(1):13–26.

[Zhang et al., 2018] Zhang, X. P., Nickels, D., Poston, R., and Dhaliwal, J. (2018). One World, Two Realities: Perception Differences between Software Developers and Testers. *Journal of Computer Information Systems*, 58(4):385–394.

[Zheng et al., 2017] Zheng, X., Julien, C., Kim, M., and Khurshid, S. (2017). Perceptions on the State of the Art in Verification and Validation in Cyber-Physical Systems. *IEEE Systems Journal*, 11(4):2614–2627.

[Zhu et al., 2006] Zhu, H., Horgan, J. R., Cheung, S. C., and Li, J. J. (2006). The First International Workshop on Automation of Software Test. In *ICSE '06: Proceedings of the 28th International Conference on Software Engineering*, pages 1028–1029. ACM Press.

[Zowghi et al., 2015] Zowghi, D., da Rimini, F., and Bano, M. (2015). Problems and Challenges of User Involvement in Software Development: An Empirical Study. In *EASE '15: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, pages 9:1–9:10. ACM Press.

[Zuccato and Kögler, 2009] Zuccato, A. and Kögler, C. (2009). Report: Functional Security Testing Closing the Software - Security Testing Gap: A Case from a Telecom Provider. In Massacci, F., Redwine, Jr., S. T., and Zannone, N., editors, *Engineering Secure Software and Systems*, volume 5429 of *Lecture Notes in Computer Science*, pages 185–194. Springer.

Appendix A

SME Size Thresholds

Table A.1 details the value thresolds, defined within [European Commission, 2015], showing the differing sizes of small and medium-sized enterprises.

Table A.1: SME size thresholds

Size	**Headcount**	**Annual Turnover**	**Annual Balance Sheet Total**
Medium	< 250	$\leq$ €50 million	$\leq$ €43 million
Small	< 50	$\leq$ €10 million	$\leq$ €10 million
Micro	< 10	$\leq$ €2 million	$\leq$ €2 million

Appendix B

Literature Review Search Strategy

Within this appendix we detail the search strategy employed when conducting the literature review and background analysis (as presented within Chapter 2). Specifically, we detail the search terms employed and the data sources used (Section B.1), as well as the adopted inclusion and exclusion criteria (Section B.2).

B.1 Search Terms and Data Sources

Given the technological context of this thesis, we structure our search terms around the activities associated with security V&V (see Section 1.3.5). These activities were determined by examining a series of software security maturity models (see Appendix M). Therefore, we target these specific activities and also employ the more encompassing terms of "V&V" and "verification and validation". Further, alongside the use of "empirical" (a search term employed by others when performing reviews of the software engineering literature e.g. [Brereton et al., 2007]), we elected to expand on this by also targetting the research methods typically employed within empirical software engineering research i.e. surveys and interviews [Daly et al., 1995, Seaman, 1999, Pfleeger and Kitchenham, 2001, Singer and Vinson, 2002, Punter et al., 2003, Hove and Anda, 2005, Cater-Steel et al., 2005, Smith et al., 2013]. Table B.1 details the search terms employed.

However, and acknowledging the inconsistencies encountered when searching (this is echoed by others e.g. [Brereton et al., 2007] highlight that it was impossible to utilise the same set of search terms across both the ACM and IEEE digital libraries; we also found that only Springer offered the ability to filter by a paper's language and

whether it was accessible based on the digital library subscription held), the search terms captured within Table B.1 were rendered as appropriate across each data source used e.g. *"security testing" AND empirical* became *(+"security testing" +empirical)* when searching within ACM and *(("security testing") AND empirical)* in IEEE. Table B.1 also captures the number of papers returned, per search term employed, before application of the defined inclusion and exclusion criteria. The devised search terms were executed through the search interfaces presented by the following digital libraries: ACM Digital Library [ACM, 2020], IEEE Xplore Digital Library [IEEE, 2020] and Springer Link [Springer, 2020].

Table B.1: Search terms employed during literature review

Search Term	**ACM**	**IEEE**	**Springer**
"security testing" AND empirical	13	28	177
"security testing" AND survey	5	13	277
"security testing" AND interviews	2	4	80
"penetration testing" AND empirical	8	6	839
"penetration testing" AND survey	7	12	812
"penetration testing" AND interviews	1	4	103
"security code review" AND empirical	0	0	3
"security code review" AND survey	0	0	3
"security code review" AND interviews	0	0	3
"security design review" AND empirical	0	0	2
"security design review" AND survey	0	0	3
"security design review" AND interviews	0	0	0
"security V&V" OR "security verification and validation"	1	5	12

As discussed within Section 2.1, whilst we started our initial literature search within the ACM, IEEE and Springer digital libraries, it was necessary - given the limited results - to broaden our search strategy and employ Google Scholar [Google, 2020] and the university's LibrarySearch [RHUL, 2020]. However, we soon encountered the issue that "initial electronic searches results in large numbers of totally irrelevant papers" [Kitchenham and Charters, 2007, p. 20]. For example, whilst the Springer numbers within Table B.1 appear "healthier", the wider set of publications encompassed by this digital library alone meant that papers within other disciplines were also being included e.g. of the 839 papers matching *"penetration testing" AND empirical*, 450 fell under "Earth Sciences" and 190 under "Engineering". It was thus important to set, and adhere to, the defined inclusion and exclusion criteria [Shafique and Labiche, 2015].

B.2 Inclusion and Exclusion Criteria

When reviewing existing literature, it is necessary to specify the inclusion and exclusion criteria used [Budgen and Brereton, 2006]. As captured by [Kitchenham and Charters, 2007], the literature selected is governed by this criteria which, in turn, is based on the research question being addressed. Given our research question and focus (see Section 1.4), we ensure that the activities associated with security V&V are captured; that the literature presents some form of empirical data; and that it also provides some level of socio-technical insight within an industrial context. Based on this, our inclusion criteria consisted of the following:

- The literature covered one or more of the activities defined within Sections 1.3.1 and 1.3.5.
- The literature provided some form of empirical data: either quantitative, qualitative or a combination of both (given the socio-technical nature of V&V, see Chapter 3, it was important to embrace all data, regardless of its underlying type).
- The literature was both in English and accessible given the terms of any digital library subscription constraints.

Our exclusion criteria consisted of:

- The literature failed to provide any form of empirical data (this was important since, as reported by [Mäntylä et al., 2015]: "we found many papers that mentioned our search terms, but lacked actual empirical data").
- The literature provided limited insight into industrial practice (given the distinction between theory and practice, see Section 2.6, it was important to ensure that the papers demonstrated actual relevance to industry practice).
- The literature provided limited insight into the socio-technical nature of the activities defined within Sections 1.3.1 and 1.3.5 i.e. the studies predominately focused on the technical (e.g. benchmarking security testing tools [Pashchenko et al., 2017]).

Aside from the specific technological focus of the research question, and the associated research objectives of this thesis, the inclusion and exclusion criteria utilised above is reflective of that which is found within other software engineering-focused literature reviews e.g. [Dybå et al., 2007, Mäntylä et al., 2015].

Appendix C

Summary of the Reviewed Empirical V&V Literature

This appendix summarises the empirical V&V literature reviewed within Chapter 2. Specifically, Table C.1 summarises the organisational and technological contexts of the studies reviewed, as well as their focus (i.e. it helps identify whether a study focuses on security V&V within UK-based software SMEs, thereby helping emphasise the gaps present within the existing literature).

Table C.1: Summary of the reviewed empirical V&V literature

Study	Organisational Context	Technological Context	Summary of Study Focus		
			UK	SMEs	Security
[Andersson and Runeson, 2002]	Swedish software organisations of varying sizes (their size is unclear given that only the number of developers within each organisation is captured).	A workshop and interviews were performed to understand the state of practice of verification and validation in industry.	×	×	×
[Baharom et al., 2005]	A variety of industry sectors (with a high percentage of small IT organisations), both government and private, of varying sizes in Malaysia.	A survey of software development practices.	×	×	×
[Basili and Selby, 1987]	Two large American organisations and an American university.	A controlled study to understand the effectiveness of various software testing strategies.	×	×	×
[Basri and O'Connor, 2010]	Small Irish software organisations meeting the given definition of: "any enterprise, organisation, department and project having up to 25 people".	Employed a survey and interviews to understand what impacts the adoption of software process standards.	×	✓	×
[Berling and Runeson, 2003]	A large Swedish organisation.	Employed a survey and interviews to understand the inspection and review of requirements specifications.	×	×	×
[Berner et al., 2005]	Unclear, but within an industrial context.	Over a three-year period, test automation is studied across 12 software projects.	×	×	×
[Brodman and Johnson, 1994]	American software organisations of varying sizes (although predominately small, their size is unclear given that only the size of the software department, within each organisation, appears to be captured).	Employed a survey, and then conducted interviews, to understand the difficulties faced by software organisations when implementing process improvement programs based on the Capability Maturity Model (CMM).	×	×	×

Table C.1: Summary of the reviewed empirical V&V literature (continued)

Study	Organisational Context	Technological Context	Summary of Study Focus		
			UK	SMEs	Security
[Causevic et al., 2009a, Causevic et al., 2009b, Causevic et al., 2010]	Unclear (presumably diverse and global).	An online survey to understand various aspects of industrial software engineering. Although the focus is on software engineering more generally, test and review-based activities are covered (including security testing).	×	×	×
[Cruzes et al., 2017]	Austrian and Norwegian software organisations of varying sizes (their size is unclear given that only the number of developers within each team is captured).	Interviews were employed to understand how security testing is performed within teams adopting Agile.	×	×	✓
[Devito Da Cunha and Greathead, 2007]	British university.	Aimed to establish whether there was a correlation between personality type and ability to perform code reviews.	✓	×	×
[Dhaliwal et al., 2011]	American software organisations of varying sizes (their size is unclear given that only the number of developers and test engineers within each organisation is captured).	An online survey examining the differences between developers and test engineers.	×	×	×
[Dunsmore et al., 2001]	British university.	An empirical investigation to understand whether the delocalised nature of object-oriented code impacts code reviews.	✓	×	×
[Dybå, 2003]	Norwegian software organisations of varying sizes (their size is unclear given that only the number of developers within each organisation is captured).	A survey was employed to understand the factors impacting software process improvement within software organisations.	×	×	×

Table C.1: Summary of the reviewed empirical V&V literature (continued)

Study	Organisational Context	Technological Context	Summary of Study Focus		
			UK	SMEs	Security
[Edmundson et al., 2013]	Unclear (developers were hired via an "outsourcing site").	An empirical study examining the effectiveness of developers when performing security code reviews.	×	×	✓
[Ellims et al., 2004]	Unclear, but believed to be a medium-sized UK-based organisation.	Three software projects were surveyed within the same organisation primarily to understand unit testing practice.	✓	×	×
[Engström and Runeson, 2010]	Unclear, but presumably Swedish software organisations of varying sizes (their size is unclear given that only the number of developers within each organisation is captured).	A qualitative survey to understand regression testing practices within industry.	×	×	×
[Epstein, 2009]	Unclear (although a medium-sized organisation is defined as having a sales volume falling between $100 million and $1 billion).	Interviews were conducted to understand how organisations perform software assurance, covering security-focused test and review-based activities.	×	×	✓
[Favre et al., 2003]	A large French software organisation.	A case study was performed to understand tool adoption issues.	×	×	×
[Garousi and Varma, 2010]	Canadian software organisations of varying sizes.	An online survey was performed to understand software testing practices in industry.	×	×	×
[Geras et al., 2004]	Canadian software organisations, presumably of varying sizes.	A survey was performed to understand software testing practices in industry.	×	×	×

Table C.1: Summary of the reviewed empirical V&V literature (continued)

Study	Organisational Context	Technological Context	Summary of Study Focus		
			UK	SMEs	Security
[Grindal et al., 2006a, Grindal et al., 2006b]	Swedish software organisations of varying sizes.	Interviews were conducted to understand software test maturity within software producing organisations. The Test Process Improvement (TPI) model was employed within this study.	×	×	×
[Groves et al., 2000]	New Zealand software organisations of varying sizes (including SMEs, but with no explicit focus).	Interviews were conducted to understand current practice regarding requirements and specifications and their verification and validation.	×	×	×
[Itkonen et al., 2009]	Finnish software organisations (the study claims to focus on SMEs, however, no definition is provided).	Involved a combination of field observations and interviews to understand system level manual test practices.	×	✓	×
[Kamsties and Lott, 1995]	German university.	A controlled experiment contrasting various approaches to defect detection.	×	×	×
[Kasurinen et al., 2010]	Software organisations, of varying sizes, all presumably based in Finland.	Employed a survey and interviews to understand software test automation practice.	×	×	×
[Kelly and Shepard, 2002]	Canadian university.	Three code inspection experiments were conducted over a period of three years.	×	×	×
[Kettunen et al., 2010]	Unclear, but presumably Finnish software organisations of varying sizes (including SMEs, but with no explicit focus).	Employs interviews to understand the differences, in terms of software testing, between software organisations adopting Agile and those adopting more traditional approaches (e.g. Waterfall).	×	×	×

Table C.1: Summary of the reviewed empirical V&V literature (continued)

Study	Organisational Context	Technological Context	Summary of Study Focus		
			UK	SMEs	Security
[Kollanus and Koskinen, 2006, Kollanus, 2009, Kollanus, 2011]	Finnish software organisations of varying sizes (however, it is unclear as to their size).	Interviews were conducted to understand current inspection practices within industry. The Inspection Capability Maturity Model (ICMM) was used to analyse the results.	×	×	×
[Koomen, 2002]	Unclear (presumably diverse and global, however, small organisations are defined as having less than 1,000 employees).	An online survey was conducted to understand the use of the Test Process Improvement (TPI) model.	×	×	×
[Larusdottir et al., 2010]	Icelandic software organisations of varying sizes (including SMEs, but with no explicit focus).	Employed an online survey, and then conducted interviews, to understand software testing practices in industry. Although the focus is usability testing, security testing is also discussed.	×	×	✓
[LaToza et al., 2006]	A large American software organisation.	Employed a survey, and then conducted interviews, to understand the working habits of developers.	×	×	×
[Mäntylä and Lassenius, 2009]	A, presumably, Finnish software organisation (its size is unclear given that only the number of developers is captured) and a Finnish university.	Two code review studies were performed (one within industry, one within academia) to determine the types of defects that are located by code review.	×	×	×
[Myers, 1978]	A large American software organisation.	A controlled experiment, utilising a survey, to contrast software testing and walkthroughs/inspections.	×	×	×
[Nagy and Víg, 2008]	Unclear (presumably diverse and global).	An online survey to understand the test approach and tools utilised on Erlang-based projects.	×	×	×

Table C.1: Summary of the reviewed empirical V&V literature (continued)

Study	Organisational Context	Technological Context	Summary of Study Focus		
			UK	SMEs	Security
[Nelson and Schumann, 2004]	The aerospace industry (believed to be American based).	A survey to determine the properties considered most important when performing code reviews. Security properties are touched upon and are considered to be of high-difficulty and of high-importance.	×	×	×
[Park et al., 2008]	Korean software defence industry (organisational size is not defined).	A survey was employed to understand software test maturity. The Test Process Improvement (TPI) model was used to assess software test maturity.	×	×	×
[Porter et al., 1998]	A large American telecommunications organisation.	A controlled experiment to understand the sources of variation in software reviews.	×	×	×
[Rodrigues et al., 2010]	Brazilian software organisations (the study claims to focus on small organisations, however, no definition is provided).	A survey is used to understand what impacts the institutionalisation of software test processes.	×	✓	×
[Rooksby et al., 2009]	Software development within various environments (e.g. industry and academia).	An ethnographic study focused on the social and organisational aspects of software testing.	×	×	×
[Runeson, 2006]	Swedish software organisations of varying sizes (including SMEs, but with no explicit focus).	An industrial survey of unit testing practice.	×	×	×
[Sawyer, 2001]	A large, global software organisation.	A survey is used to understand intra-group conflict within a software development team.	×	×	×
[Sitnikova et al., 2007]	South Australian software defence industry (including SMEs, but with no explicit focus).	Interviews were performed to understand the software engineering processes, standards and tools used.	×	×	×

Table C.1: Summary of the reviewed empirical V&V literature (continued)

Study	Organisational Context	Technological Context	Summary of Study Focus		
			UK	SMEs	Security
[Staples et al., 2007]	Australian software organisations of varying sizes.	An empirical study examining the reasons why software organisations do not adopt the Capability Maturity Model Integration (CMMI) model.	×	×	×
[Sung and Paynter, 2006]	New Zealand software organisations of varying sizes.	An online survey to understand software testing practices in industry.	×	×	×
[Tervonen and Harjumaa, 2004]	Finnish software organisations (the study claims to focus on SMEs, however, no definition is provided and it is unclear whether SME-sized departments are within SME-sized organisations).	An examination of the motivational factors, and obstacles, to adopting software reviews and inspections.	×	✓	×
[Thörn and Gustafsson, 2008]	Swedish software SMEs (as defined within [European Commission, 2015]).	A survey to understand software modelling tools and techniques.	×	✓	×
[Torkar and Mankefors, 2003]	Unclear (but includes Swedish and American organisations of varying sizes).	A survey focused on software testing and reuse.	×	×	×
[Uwano et al., 2008]	Japanese university.	An experiment to determine the differences in subject performance when undertaking design and code reviews.	×	×	×
[Wicks, 2005]	British software organisations of varying sizes (∼72% employ less than 100 people).	Employs a survey to examine the tools utilised in software development.	✓	×	×
[Wojcicki and Strooper, 2006]	Unclear (presumably diverse and global).	A survey to understand the state of practice regarding the verification and validation of concurrent programs.	×	×	×
[Wood et al., 1997]	British university.	An empirical study comparing the effectiveness, and efficiency, of software testing and code reading.	✓	×	×

Table C.1: Summary of the reviewed empirical V&V literature (continued)

Study	**Organisational Context**	**Technological Context**	**Summary of Study Focus**		
			UK	**SMEs**	**Security**
[Zelkowitz et al., 1984]	American and Japanese software organisations (their size is unclear).	Employed a survey, and then conducted interviews, to understand software engineering practice.	×	×	×
[Zhang et al., 2018]	Unclear (presumably diverse and global).	An online survey examining the differences between developers and test engineers.	×	×	×
[Zheng et al., 2017]	Unclear, but global and presumably of varying sizes.	Employed an online survey, and then conducted interviews, to understand the state of practice of verification and validation in industry.	×	×	×

Appendix D

Question Themes

Within this appendix we present a series of question themes, structured around the key themes of STIN (see Section 3.4), in order to address the research objectives.

D.1 Addressing Research Objectives 1 and 3

Within Sections 4.5.1 and 4.5.2 we indicated the models used to derive a series of question themes (QT) to address Research Objectives 1 and 3. These are presented and discussed below.

QT1: *Which V&V activities are performed within software SMEs?*

We begin by establishing whether the activities encompassed by our adopted definitions of software and security V&V (see Sections 1.3.1 and 1.3.5 respectively) are performed. These activities are covered, to varying degrees, in SAMM, BSIMM, TPI, BDTPI, TMMi and ICMM. Similar to [Larusdottir et al., 2010], we also aim to understand the extent an activity is employed.

In addition, we intend to understand at what point in the software development lifecycle the V&V activities are performed (which is reflected within both TPI and BDTPI, where earlier involvement suggests a higher level of maturity). We also intend to understand whether the security testing performed is functional (BSIMM: ST1.3), risk-based (BSIMM: ST3.3), or both. We view organisations practising some form of risk-based testing as being more mature (due to the increased complexity e.g. [Potter and McGraw, 2004]).

QT2: *Why are the V&V activities performed?*

Having established whether an activity is performed, we then intend to understand the incentives and motivations for performing the activity. Runeson used

a survey to examine developer motivation when performing unit testing by asking developers to indicate "very good", "good", "neutral", "bad", "very bad" or "not applicable" [Runeson, 2006] and Kollanus and Koskinen utilised interviews to determine the motivation around conducting reviews [Kollanus and Koskinen, 2006]. BSIMM indicates that code reviews should become mandatory for all projects (BSIMM: CR1.5). With regards to STIN, this will enable an understanding to develop in terms of an interactors' incentives and their motivation for performing an activity.

QT3: *Who performs and owns the V&V activities?*

It is important to establish who is responsible for performing and owning each activity (especially when recalling that people are more likely to wear multiple hats within SMEs [Murthy, 2009, Cruz-Cunha, 2010, Linares et al., 2018], as well as the views surrounding test-based activities, see Section 2.6). For example, the TPI model views unit testing and integration testing as being predominately performed by developers [Koomen and Pol, 1999]. Whilst developers are generally associated with performing unit testing [Andersson and Runeson, 2002], they reportedly lack motivation in this regard [Runeson, 2006] and typically perform a limited set of positives tests [Hunt et al., 2007]. The TMMi model stresses the importance of an independent test group [TMMi Foundation, 2018], thus indicating developer and test engineer independence. The concept of assigning responsibility is covered heavily within TMMi. Within BSIMM there is reference to having either an organisation's software security group leading the code or design reviews (BSIMM: CR1.2 and AA1.3) or software architects (BSIMM: AA3.1). In addition, we seek to understand whether external resources are relied upon e.g. for penetration testing (BSIMM: PT1.1 and PT3.1). Larusdottir et al. aimed to understand who performed security testing, but not security design and code reviews [Larusdottir et al., 2010]. In the context of STIN, this will help establish the interactors, the activities they perform, as well as where any potential signs of conflict exist.

QT4: *Who should perform and own the V&V activities?*

We aim to understand where a respondent believes responsibility for the activities should reside. This, in terms of STIN (and in conjunction with QT3), will enable the interactors excluded from the V&V activities to be identified.

QT5: *Are automation technologies and tools used to support the V&V activities?*

The value of test automation, and the use of tools, was discussed in Section 2.4.

Their adoption is generally encouraged throughout BSIMM and SAMM e.g. using automated tools alongside manual review (BSIMM: CR1.4); incorporating black-box security tools into the quality assurance process (BSIMM: ST2.1); automating security tests (BSIMM: ST2.5, SAMM: IR3.1 and SAMM: ST2.1) and using penetration testing tools in-house (BSIMM: PT1.3). Unsurprisingly, one of the key areas of focus of the TPI model concerns test automation, and whilst TMMi does not have a specific area of focus regarding tools and automation, where such technology can provide support, this is detailed e.g. the use of performance testing tools under non-functional testing.

Aside from understanding where automation is applied, and to what extent, we also intend to understand whether tools are utilised (e.g. are automated code analysis tools to find security problems accessible (SAMM: IR2.1)) and whether they are developed in-house, are proprietary, are open source, or a combination of these (there is an indication that SMEs utilise open source tools to reduce software development costs [Sitnikova et al., 2007], however, in terms of V&V-related tools this is unclear). In addition, we aim to understand whether they are customised to an organisation's environment (customisation, within BSIMM for example, is a sign of higher maturity e.g. customising penetration testing tools and scripts (BSIMM: PT3.2), using automated tools with tailored rules (BSIMM: CR2.6) and customised fuzz testing (BSIMM: ST2.6)). In terms of STIN, this will assist in the identification of non-human interactors.

QT6: *How are the results from the V&V activities communicated?*

Successful V&V requires good interactor communication and collaboration (see Section 3.2). Both BSIMM and SAMM focus on communicating the results from the security V&V activities performed e.g. for security testing (BSIMM: ST2.4, SAMM: ST1.3 and SAMM: ST2.2) and security code reviews (SAMM: IR2.2); as well as ensuring that the results are captured within a defect tracking system (BSIMM: PT1.2) - with such centralised reporting being used "to close the knowledge loop and drive training" (BSIMM: CR1.6). The TPI model also covers communication and reporting (with maturity tied to the different types of information being reported) and emphasises the importance of not only communicating the results, but the communication between different types of interactors e.g. test engineers, developers and customers. These are also covered within BDTPI and TMMi. Therefore, we intend to understand not only which activities are communicated, and through what mechanism, but also to who the information is circulated. In the context of STIN, this involves understanding the communica-

tion forums in place, as well as the relevant interactors and who they interact with.

QT7: *Do the V&V activities have a governing process?*

A defined and documented process typically denotes a more mature organisation (e.g. the CMMI model [CMMI Product Team, 2010]). Both BSIMM and SAMM aim to establish whether processes exist for some of the security V&V activities e.g. for undertaking, and requesting, security design reviews (BSIMM: AA2.1 and SAMM: DR2.2 respectively), as well as following a consistent process when performing and reporting on security testing (SAMM: ST2.2). The TMMi model also views organisations with defined and documented processes as being more mature where, at the lowest level, "testing is a chaotic, undefined process" [TMMi Foundation, 2018, p. 10]. Therefore, we intend to establish, if a process exists, whether this is based upon a process generated in-house or upon an existing standard. We also intend to understand whether the organisation is undergoing any form of process improvement and, if so, which approach they are utilising e.g. BSIMM or TPI, as well as whether they are familiar with these (Rodrigues et al., within seven small organisations (however, "small" is not defined within their study), reported on the use, and knowledge of the existence of, two test process models - including TPI [Rodrigues et al., 2010]). This enables non-human interactors, in the form of processes, to be identified and adds additional support as to whether their presence acts as an incentive to perform a V&V activity.

QT8: *Are the V&V activities performed well? What could be improved?*

We aim to understand the respondent's perception of how the activities associated with software and security V&V are performed within their organisation. Dependent on the response received, it will either permit an understanding to develop regarding the respondent's capabilities, and thus whether the respondent should be viewed as a credible authority, or it will lead, in a STIN context, to uncovering the undesired interactions (and possibly resource flows) which surround the activities.

QT9: *How much is spent on the V&V activities?*

Understanding the proportion of a project's budget dedicated to the activities of software and security V&V is necessary, in the context of STIN, for understanding resource flows i.e. by "following the money" [Kling et al., 2003, p. 58] (it also helps identify the value an organisation places on a particular activity). The concept of a budget is covered in the BDTPI model (BDTPI: Stakeholder Commitment,

Level C, Checkpoint 2) and the TPI model (under the area of "commitment and motivation", which includes the assignment of budget and time to testing).

QT10: *Should there be more investment in the V&V activities?*

Following on from QT9, we aim to understand whether the respondent believes more investment is required in software and security V&V. Without suitable levels of investment, the activities will, of course, suffer.

QT11: *What impacts the V&V activities?*

To identify what impacts different aspects of testing (including security testing, but not penetration testing), Larusdottir et al. provided the following options: "Lack of training/knowledge", "Lack of budget", "Lack of time", "Other" and "N/A" [Larusdottir et al., 2010]. These options should also include lack of resource and management support (which are issues which have resulted in information security challenges within SMEs e.g. [Dojkovski et al., 2006, Dojkovski et al., 2007, Williams, 2009]). It is also sensible to understand whether an activity is deemed "dispensable", since test-based activities have previously been viewed in such a light e.g. [Nguyen et al., 2006, Rodrigues et al., 2010] (Rodrigues et al. include this, and a lack of tool support, when examining the barriers to test process implementation within small organisations). In addition, we intend to understand which aspects of V&V are most likely to get squeezed when a project faces a shortage of time (Torkar and Mankefors posed such a question when conducting a survey on testing and code reuse, but just consider V&V in its entirety and not by individual activity [Torkar and Mankefors, 2003]). In terms of STIN, this will lead to understanding the undesired interactions faced within an organisation.

QT12: *What V&V related training has been received?*

Within Chapter 1 we indicated that software engineering is complex and that finding vulnerabilities is difficult [Austin and Williams, 2011]. However, whilst other studies have aimed to identify whether training is provided e.g. [Baharom et al., 2005, Wicks, 2005, Kollanus and Koskinen, 2006, Garousi and Varma, 2010], the focus is typically on software engineering more generally, or software V&V - there is relatively little information regarding security V&V training, especially within an SME context. The "correct composition of a test team is very important" [Koomen and Pol, 1999, p. 38] (both technical and social skills are required, see Section 3.2), therefore, we also intend to understand whether

training has been provided across the activities associated with software V&V (BDTPI: Tester Professionalism, Level C, Checkpoint 1).

In addition, we intend to understand through what medium the training has been received e.g. formal training courses, mentoring (which is touched upon in BSIMM: AA2.3 and CR2.5), reading or conference attendance (BDTPI: Tester Professionalism, Level O, Checkpoint 1), as well as how often and when such training was last provided. Aspects of training are also covered within TMMi and ICMM. Whilst expertise, in a STIN context, is considered a resource flow [Meyer, 2007, Taylor-Smith, 2016], understanding V&V training within SMEs will also help support findings regarding undesired interactions i.e. the inability to perform an activity due to a lack of training/knowledge.

The above twelve question themes were developed through reference to a number of software V&V and software security maturity models, namely: [Koomen and Pol, 1999, de Vries et al., 2009, Kollanus, 2011, McGraw et al., 2016, OWASP, 2017, TMMi Foundation, 2018] and to a number of existing, relevant studies e.g. [Runeson, 2006, Kollanus and Koskinen, 2006, Larusdottir et al., 2010, Rodrigues et al., 2010]. We note that the sole adoption of any one of these models, or existing studies, would have been insufficient in terms of addressing the research objectives of this thesis. For example, either by not encompassing all of the V&V activities as defined within Sections 1.3.1 and 1.3.5, or by overlooking important aspects (e.g. although both SAMM and BSIMM touch upon automation, there is no understanding to what extent it is employed within an organisation). Within Appendix H we present the resulting survey questions, formulated around these question themes and, within Appendix K, we present the corresponding interview guide.

D.2 Addressing Research Objective 2

Examining a number of the relevant questions stated within [BIAC and ICC, 2003], we devise a series of question themes (QT) in order to assess information security culture within SMEs. These are presented below and are supported through reference to other relevant information security culture studies.

QT1: *Do you have written information security policies that everyone knows and understands?*

The possession of an information security policy is considered fundamental [von Solms, 2001b] - certainly, in terms of an organisation having a basis on which

to build an information security culture. Small organisations should have a relevant, "simple and clear" information security policy, which is distributed to all employees [BIAC and ICC, 2004, p. 21]. However, we extend the question as found within [BIAC and ICC, 2003] to capture both whether the respondent is aware of such a policy themselves and whether the respondent believes that their organisation's employees, in general, are aware of the policy's existence. If a respondent is unable to indicate whether such a policy exists, it is clear that the organisation does not have a strong basis for developing an information security culture. If a respondent cannot provide a definitive answer i.e. "yes" or "no" to such a question we, like in [Dimopoulos et al., 2004], interpret this as being effectively equivalent to a "no", since: "even if the organizations concerned do actually have a policy, they are evidently not promoting it to their staff in an *[sic]* successful manner" [Dimopoulos et al., 2004, p. 79]. Therefore, understanding how updates to the information security policy are communicated to the organisation's employees should form a separate question (in the context of STIN, this concerns the identification of existing communication forums). Further, we intend to differentiate between knowledge of an information security policy's existence and knowledge of the actual content. We also observe that senior management should visibly support an organisation's information security policy [Thomson et al., 2006] - QT4 covers management responsibility - therefore, we include this as an additional question which, in the context of STIN, will help identify incentives and resource flows.

QT2: *Are your personnel security aware and security educated?*

There is a need, within organisations, to foster a security culture whereupon employees are aware of the issues that are relevant to them, as well as enabling them to act appropriately - this can be achieved through training and education [Furnell, 2007]. Awareness is a "cornerstone of a security culture" [Williams, 2009, p. 52], with security training and education considered essential in developing and maintaining such a culture. For example, Kruger and Kearney indicate that the "key to success in awareness is keeping the messages relevant and consistent" [Kruger and Kearney, 2006, p. 290] - thus implying that the development and maintenance of an information security culture is ongoing. Educating employees, with regards to their roles and responsibilities, requires continual reinforcement when it comes to security [Ruighaver et al., 2007] (QT5 concerns roles and responsibilities). We modify the question - as originally stated within [BIAC and ICC, 2003] - to elicit a response in terms of an individual, and the individual's

belief across their organisation as a whole (this approach has previously been adopted when assessing information security culture e.g. [Schlienger and Teufel, 2003, Ngo et al., 2009]). In addition, it is sensible to understand whether training is performed only during a respondent's induction or on a more continual basis (too many organisations only provide limited induction training [Ruighaver et al., 2007] - unfortunately, Ruighaver et al. do not elaborate upon the type of organisations or industry sectors in this regard). However, according to [BIAC and ICC, 2003], everyone "should understand their roles and responsibilities in relation to security assurance, as well as those of the people they deal with" - this implies that an individual should understand whether an information security function exists within their organisation - this is covered by QT4.

QT3: *What do you do to raise security awareness across your partners, suppliers and users?*

This question, as it stands within [BIAC and ICC, 2003], is modified to ask: "What have you done, or do, to raise security awareness within your organisation"? Understanding what a respondent does, or has done, to increase security awareness, within an organisation, will provide some indication as to whether an individual has embraced security-related practices. Namely, if a respondent has actively participated in raising the level of information security awareness within their organisation, we would consider this indication that they have started to move to the "competent" stages on Thomson and von Solms Information Security Competence Maturity Model [Thomson and von Solms, 2006]. As noted by Talib et al.: "simply undertaking training or having an awareness of an issue does not necessarily imply practice" [Talib et al., 2010, p. 200]. Further, understanding what a respondent's organisation does to raise security awareness forms a secondary component to this question which will, in the context of STIN, also help identify incentives and resource flows.

QT4: *Do you have an information security function (person or group) that reports to senior management, such as the Board or executive committee?*

Ensuring that information security has representation that reports to an organisation's senior management is essential. Observing that an organisation's information is critical to its success, von Solms indicates that as an organisation's board of directors is accountable for the organisation's success they are, therefore, responsible for ensuring information security [von Solms, 2001b]. The role of senior management, in terms of developing and maintaining an organisation's information security culture, cannot be overstated (we also recall that senior

management can influence V&V within an organisation [Perry, 2006]). For example, information security is considered, in general, a management problem, therefore, the "security culture reflects how management handles this problem" [Ruighaver et al., 2007, p. 56]. Ruighaver et al. state that an organisation's senior management has responsibility to support and embed information security within an organisation. In addition, we explore whether an individual, or group of individuals, are responsible for information security and whether this is a shared responsibility across the organisation's employees or not. Dimopoulos et al. found, within a survey of SMEs, that "the majority of organisations do employ someone who is assigned this task, and the proportion increases with the size of the organisation" [Dimopoulos et al., 2004, p. 76]. In the context of STIN, these questions will help identify the interactors involved.

QT5: *Are your employees aware of their responsibilities to help maintain security?*

It is important to ensure that an organisation's employees are both dedicated and knowledgeable with regards to their roles and responsibilities when it comes to information security [Thomson and von Solms, 2006]. Thomson and von Solms conclude that this "could be the strongest link in the information security infrastructure". Failure to ensure this understanding can result in an organisation unable to protect the confidentiality, integrity and availability of its information assets [Thomson et al., 2006] (see Section 1.3.3 for a discussion of these terms). Notably, even if an organisation has implemented an information security awareness campaign, this does not imply that all employees understand their role in terms of ensuring information security [Kruger and Kearney, 2006]. Further, it has previously been reported that "[s]ecurity can become to be seen as the role of 'others' rather than the role of all members of the organisation" [Gokhale and Banks, 2004, p. 178]. However, this is another question that should be posed in the context of an individual, and in terms of the individual's view across their organisation. Employee responsibility does not remove the need for senior management support [Ruighaver et al., 2007] (QT4 concerns management support), however, although responsibility is typically associated with layers of management, to become part of an organisation's "cultural fabric", responsibility must be adopted by all employees [Williams, 2009, p. 51]. Failure to appropriately assign responsibility, within SMEs, could lead to "serious difficulties if an incident occurred, as there would be no clear point of contact" [Dimopoulos et al., 2004, p. 76]. That some SMEs appear to include acceptance of responsibility within an employee's contract of employment (thus stipulating that an employee must

follow the organisation's information security policy) e.g. [Burns et al., 2006], we include this as a separate question, since it implies that the organisation's management is actively "encouraging" employees to follow the organisation's information security policy. Such an approach can influence employee motivation [Dojkovski et al., 2007] which, in a STIN context, relates to identifying incentives and resource flows.

The above five question themes were devised around the questions within [BIAC and ICC, 2003] - which are relevant to SMEs [BIAC and ICC, 2004] - and were expanded upon based on a number of existing, information security culture studies e.g. [Schlienger and Teufel, 2003, Burns et al., 2006, Ngo et al., 2009]. Appendix H presents the survey questions, developed from these question themes, with the associated interview guide detailed within Appendix K.

Appendix E

Survey Instrument Pre-Testing

This appendix details the approach (Section E.1), the respondents (Section E.2) and the survey instrument pre-testing sessions conducted (Section E.3).

E.1 A Concurrent and Retrospective Approach

As captured within Section 5.1.1, we employed two approaches when pre-testing the survey instrument:

- A concurrent approach [Drennan, 2003]: where we sat with a respondent whilst they completed the survey and encouraged them to think aloud. Utilised during cognitive interviewing, respondents verbalised their thought processes as they completed the survey [DeMaio et al., 1998]. This allowed prompting when either a problem was observed, or when a decision had been made (notably, such personal interviews allow reactions and hesitations to be identified [Hunt et al., 1982]). We also employed paraphrasing i.e. respondents were asked to paraphrase certain survey questions in order to help remove comprehension problems [Collins, 2003].
- A retrospective approach [Drennan, 2003]: where a respondent completed the survey in their own time, by themselves, with the simple advice of recording any comments or concerns as they went. This helped reduce any bias being introduced by being present during survey completion. On completion, the comments captured and circulated by the respondent were discussed. This approach also ensured that different online environments were utilised whilst completing the survey and ultimately was more closely aligned to when the survey went live [DeMaio et al., 1998].

Although the former approach enables information to be collected as it first becomes

available, it might lead to bias [DeMaio et al., 1998]. However, the latter approach provides an unbiased means of collecting information - but at a point where a respondent's thought processes might not be accurately recalled [DeMaio et al., 1998] (it also helps address the issue of respondents being unable to verbalise their thought processes as they completed the survey [Drennan, 2003]). Clearly, there is a trade-off between both approaches, however, they can be combined [Drennan, 2003]. Notably, when following a personal interview approach, subsequent pre-testing "should be conducted by means of the administration method to be used in the ultimate research" [Hunt et al., 1982, p. 270]. Therefore, employing the two approaches in combination was sensible. In addition, both approaches may result in different types of information being collected [DeMaio et al., 1998].

E.2 Respondent Profiles

As respondents should be purposively identified [DeMaio et al., 1998], we highlight that the subject matter experts involved were all practising software engineers, based within industry, and who had performed V&V activities on average for at least eight years within both SMEs and large organisations. Noting that respondents should "match the characteristics of the proposed sample" [Drennan, 2003, p. 59], this helped ensure that both the relevant technical and organisational nuances of the survey had been tested by an appropriate set of respondents. Also, three of the respondents had English as a second language, which further helped ensure question accessibility. Finally, we revisited the survey with the first two respondents (respondents 1 and 2 during phases three and four respectively) to ensure that their comments had been addressed to their satisfaction, and that changes resulting from the feedback of subsequent sessions had not adversely affected the instrument. This was an important step, since it is noted that revisions resulting from instrument pre-testing are not usually retested [DeMaio et al., 1998]. Table E.1 details the respondents involved in the survey instrument pre-testing.

Table E.1: Survey instrument pre-testing: respondent profile and relevance

ID	Type of Respondent	Technical Relevance	Organisational Relevance				Language (English)	Experience (Years)
			Very Small	Small	Medium	Large		
1	Subject matter expert	V&V practitioner	×	✓	✓	✓	First	6 - 10
2	Subject matter expert	V&V practitioner	✓	×	×	✓	First	6 - 10
3	Subject matter expert	V&V practitioner	×	×	✓	✓	Second	20+
4	Technical expert	Former software practitioner turned academic	✓	×	×	×	First	11 - 20
5	Subject matter expert	V&V practitioner	×	✓	✓	✓	Second	6 - 10
6	Subject matter expert	V&V practitioner	✓	×	×	✓	First	6 - 10
7	Subject matter expert	Former V&V practitioner turned developer	×	✓	✓	✓	Second	6 - 10
8	Technical expert	IT engineer	×	✓	✓	✓	First	20+
9	Subject matter expert	V&V practitioner	×	×	×	✓	First	6 - 10
10	Subject matter expert	V&V practitioner	×	✓	×	✓	First	3 - 5
11	Technical expert	Software technical author	×	×	✓	✓	First	11 - 20

E.3 Pre-Testing Session Information

The pre-testing sessions were conducted over four phases:

- We began by sitting with a subject matter expert and walking through the survey. Adopting a think aloud approach, this was a very useful session, resulting in a number of revisions simplifying the survey and ensuring that the questions had an appropriate set of options. Of the 16 comments received, 11 were implemented.
- The second phase involved sitting with another subject matter expert and walking through the revised survey. Once again, a think aloud approach was adopted. Sixteen comments were received, 13 of which were implemented. These resulted in additional options being added to a number of questions, as well as a few changes to the survey's appearance.
- During the third phase, three sessions (two retrospective, one think aloud) were conducted with subject matter experts and one session (retrospective) with a former software practitioner turned academic who has an interest in surveys. Collectively, the four sessions resulted in 29 comments, 6 of which were addressed.
- Within the fourth phase, five sessions (three think aloud walkthroughs and two retrospective sessions) were conducted and involved four subject matter experts and one technical author (who has worked in the software industry for over 10 years). In total, 72 comments were received, 42 of which were addressed. We highlight that 34 of those addressed resulted from the session held with the technical author, the majority of which concerned minor improvements to the survey text.

After spending just over 15 hours sitting with the respondents, a total of 133 comments were received, 72 of which were addressed. The comments not addressed typically fell into the following categories: the respondents proposed new options or questions, or sought some form of clarification. In all cases we ensured that an understanding was reached with the respondent before discounting any comment. In addition, many positive comments were received that required no further action, or were identified as duplicates. This feedback was obtained following six concurrent and five retrospective pre-testing sessions. We observe that the number of cognitive interviews required when pre-testing a survey is small (typically 20 or under) [DeMaio et al., 1998]. DeMaio et al. also indicate that respondents should be purposively identified. As an example, Wildy and Clarke opted for five cognitive interviews when utilising a convenience

sample [Wildy and Clarke, 2009]. Table E.2 details the survey instrument pre-testing sessions conducted.

Table E.2: Survey instrument pre-testing: session information

ID	**Method Employed**	**Session Length (Hours)**	**Comments**		**Survey Completion Time (Minutes)**
			Raised	**Addressed**	
1	Concurrent	1	16	11	30
2	Concurrent	1.5	16	13	-
3	Retrospective	1	10	1	-
4	Retrospective	0.75	7	1	25
5	Retrospective	1.5	5	1	20 - 30
6	Concurrent	1	7	3	20 - 30
7	Retrospective	1.5	3	0	25
8	Retrospective	1	4	0	35
9	Concurrent	0.75	5	1	-
10	Concurrent	2	25	7	-
11	Concurrent	3	35	34	-

Appendix F

FAME Search Strategy

The Financial Analysis Made Easy (FAME) database [Bureau van Dijk, 2020] was used to derive our study sample (such an approach is consistent with other SME-focused studies e.g. [Duan et al., 2002, Wong and Aspinwall, 2005, Martínez-Caro and Cegarra-Navarro, 2010, Chang et al., 2011, Sinkovics et al., 2013, Ramdani et al., 2013, Dyerson et al., 2016, Mawson and Brown, 2017]). Therefore, based upon the defined organisational scope (see Section 1.2), we formulated our search strategy as follows:

- Firstly, we restricted by industry sector. Within Section 1.2.2, we defined software enterprises as organisations falling under SIC class 62.01 (specifically, encompassing the two sub-classes 62.01/1 and 62.01/2).
- Secondly, we restricted by number of employees and turnover. Based on our adopted SME definition (i.e. [European Commission, 2015], see Section 1.2.1), we stipulated that an organisation must not exceed 249 employees and a turnover of €50 million. We disregarded any organisations exceeding these upper bounds (i.e. we did not utilise the "use estimates" option available within FAME). We observe that only ∼3.6% of the active companies listed within FAME detail the number of employees (similar has been observed e.g. [Rogers et al., 2007]).

Table F.1 details the employed FAME search strategy.

Table F.1: FAME search strategy

Search Step	**Step Result**	**Search Result**
1. All active companies (not in receivership nor dormant) and companies with unknown situation	3,212,721	3,212,721
2. UK SIC (2007): All codes: 6201 - Computer programming activities, 62011 - Ready-made interactive leisure and entertainment software development, 62012 - Business and domestic software development	39,986	30,807
3. Number of Employees: Last available year, max=249	190,530	737
4. Turnover (th EUR): Last available year, max=50,000	609,950	682
Boolean search : 1 And 2 And 3 And 4	TOTAL	682

Appendix G

Respondent Communication

Communication with the respondents was predominately based on the templates presented within Sections G.1 and G.2. These adhere to the recommendations detailed within Sections 4.3.1 and 5.1.3. For example:

- We indicated that through completing the survey researchers' would be better placed to improve upon current V&V practice. Smith et al. found, when surveying software developers, that higher response rates may be achieved when indicating a benefit to society [Smith et al., 2013].
- University affiliation was stated so as to appear as a credible, legitimate authority - thereby helping engender trust [Singh et al., 2009] - which, in turn, helped when providing assurance that any published data (the publishing of data is also considered as benefiting society [Smith et al., 2013]) would "only be presented in non-identifiable statistical format".
- We indicated, on survey completion, that a report could be made available showing how the respondent's organisation compared with similar organisations. Observing the importance of motivating respondents, Punter et al. adopt such an approach [Punter et al., 2003].

In some instances, especially in terms of the interviews, communication was much more "free-form" and, alongside emails simply expressing our gratitude for participating in the study and acquiring a respondent's telephone number for example, are not detailed here.

G.1 Initial Contact Template

Dear <*Title and Name*>,

Please allow me to introduce myself: I am a researcher, based at Royal Holloway, University of London and I am exploring how small and medium sized organisations perform software verification and validation both in general, and when focussing on security.

As the <*Position*> at <*Company*> (and as someone <*Why respondent was selected*>), this project would be greatly aided by understanding your views on this topic. As such, would you kindly be willing to help with this research by participating in a short online survey (please note that all collected data would only be presented in non-identifiable statistical format)?

Your participation will help address a gap in our knowledge, thereby enabling researchers to better improve upon current practice. It would also enable you to see how your organisation compares with similar organisations.

Thank you for your time and, hopefully, your help - it is greatly appreciated.

I look forward to hearing from you.

With best wishes,

Matthew Kreeger
Technology and Information Management Research Group
Royal Holloway, University of London

P.S. If you feel that someone else is perhaps better placed to complete the survey, would you kindly be able to suggest who I should contact within your organisation?

G.2 Subsequent Contact Template

Dear *<Title and Name>*,

Thank you for your email. The survey can be accessed via: *<Unique Qualtrics link>*.

<Address any specific questions from respondent>. If you are kindly able to complete the survey by the *<Date>* that would be much appreciated. Please note that you can revisit the survey as time allows as your progress will be saved.

Once again, thank you - I greatly appreciate your taking the time to help with this project.

With best wishes,

Matthew Kreeger
Technology and Information Management Research Group
Royal Holloway, University of London

Appendix H

Survey Instrument

The survey instrument used for the first phase of data collection (see Chapter 5) is presented within this appendix (with the questions being devised from the question themes presented in Appendix D). The survey, which was developed and distributed through the use of Qualtrics [Qualtrics, 2020], underwent a series of pre-testing (see Section 5.1.1) and was generated following the design considerations detailed within Chapter 4. It is sensible to note, however, that this appendix presents the survey in the form of static text and thus does not capture the dynamic and interactive elements offered by the survey, for example: it does not reflect the underlying display logic (which determines the questions to pose to a respondent based upon their answers as they progress through the survey) and nor does it convey how the questionnaire is displayed to a respondent when utilising elements such as drop-down menus and sliders. In addition, and to avoid repetition, each page presented to a respondent contained the following:

- A header, which encompassed the university logo in order to convey authority and to build trust (essential for when conducting research into a sensitive subject).
- A footer, which included a back and next button as appropriate, as well as a progress bar.

We now present the survey instrument.

H.1 Front Page

Software Verification and Validation Practice within SMEs

The purpose of this survey is to understand how SMEs perform software verification and validation both in general, and when focussing on security.

To the best of our knowledge, this is the first study of its kind that specifically focuses on organisations such as your own, when studying this critical area of software development.

Survey structure:

This survey is made up of three sections:

- A software verification and validation practice section: which is where we aim to build up a picture of how software verification and validation is carried out within your organisation.

- An information security culture section: which is where we aim to understand what type of information security culture exists within your organisation.

- A set of demographic questions.

There is a minimum of 30 questions and a maximum of 62 depending on your responses. This should require no more than approximately 12 - 26 minutes of your time to complete (please note that if you leave the survey at any point your progress will be saved, it is then possible to return to the survey).

This study has received ethical approval by Royal Holloway, University of London. All survey responses are anonymous, and the collected data will be presented in non-identifiable statistical format.

How your participation helps:

Firstly, your feedback will help contribute much needed data about a gap in our empirical knowledge. By virtue of this, researchers will be significantly better placed to improve upon current levels of software verification and validation practice.

Secondly, it will provide you and your organisation with the opportunity to see how your organisation compares with similar organisations.

Thank you very much for your participation and for helping contribute to an important area of research.

Please click the "NEXT" button to begin this section.

H.2 Section One: Software Verification and Validation Practice

Section 1: Software Verification and Validation Practice

In this section we aim to build up a picture of how software verification and validation is carried out within your organisation.

Please click the "NEXT" button to begin this section.

Section 1: Which of the following activities are carried out?

Question 1: Please indicate the extent to which the following activities are currently carried out within your organisation.

For software verification and validation related activities in general:

	Always	Often	Sometimes	Rarely	Never	Do Not Know
Software Design Reviews	○	○	○	○	○	○
Software Code Reviews	○	○	○	○	○	○
Software Testing	○	○	○	○	○	○

For security-based software verification and validation activities:

	Always	Often	Sometimes	Rarely	Never	Do Not Know
Security Design Reviews	○	○	○	○	○	○
Security Code Reviews	○	○	○	○	○	○
Security Testing	○	○	○	○	○	○
Penetration Testing	○	○	○	○	○	○

Section 1: Why are these activities carried out?

Question 2: Why are the following activities carried out within your organisation? (*please select all that apply*)

For software verification and validation related activities in general:

	To improve software quality	Mandated by process	Instructed by management	Requested by customers	To tick a box	Because it is fun	Other
Software Design Reviews	☐	☐	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐	☐	☐

For security-based software verification and validation activities:

	To improve software quality	Mandated by process	Instructed by management	Requested by customers	To tick a box	Because it is fun	Other
Security Design Reviews	☐	☐	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐	☐	☐

Question 3 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Why are these activities not carried out?

Question 4: Why are the following activities not carried out within your organisation? (*please select all that apply*)

For software verification and validation related activities in general:

	Lack of training / knowledge	Lack of budget	Lack of time	Lack of people	Lack of management support	Lack of tool support	Lack of motivation	Activity not required	Other
Software Design Reviews	☐	☐	☐	☐	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐	☐	☐	☐	☐

For security-based software verification and validation activities:

	Lack of training / knowledge	Lack of budget	Lack of time	Lack of people	Lack of management support	Lack of tool support	Lack of motivation	Activity not required	Other
Security Design Reviews	☐	☐	☐	☐	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐	☐	☐	☐	☐

Question 5 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Who performs and owns these activities?

Question 6: Who performs and who owns the following activities within your organisation? (*please select all that apply*)

For software verification and validation related activities in general:

	Who Performs					Who Owns
	Development	Test	Outsourced Engineers	Customer	Other	
Software Design Reviews	☐	☐	☐	☐	☐	
Software Code Reviews	☐	☐	☐	☐	☐	
Software Testing	☐	☐	☐	☐	☐	

For security-based software verification and validation activities:

	Who Performs					Who Owns
	Development	Test	Outsourced Engineers	Customer	Other	
Security Design Reviews	☐	☐	☐	☐	☐	
Security Code Reviews	☐	☐	☐	☐	☐	
Security Testing	☐	☐	☐	☐	☐	
Penetration Testing	☐	☐	☐	☐	☐	

Question 7 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Who should perform and own these activities?

Question 8: Regardless of whether the activities are carried out within your organisation, who do you believe should perform and who should own the following activities within your organisation? (*please select all that apply*)

For software verification and validation related activities in general:

	Who Should Perform					Who Should Own
	Development	Test	Outsourced Engineers	Customer	Other	
Software Design Reviews	☐	☐	☐	☐	☐	
Software Code Reviews	☐	☐	☐	☐	☐	
Software Testing	☐	☐	☐	☐	☐	

For security-based software verification and validation activities:

	Who Should Perform					Who Should Own
	Development	Test	Outsourced Engineers	Customer	Other	
Security Design Reviews	☐	☐	☐	☐	☐	
Security Code Reviews	☐	☐	☐	☐	☐	
Security Testing	☐	☐	☐	☐	☐	
Penetration Testing	☐	☐	☐	☐	☐	

Question 9 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Do you utilise tools with these activities?

Question 10: Are tools (for example, static code analysis tools, code review systems, test management software, test code generators and so on) used to help with the following activities? (*please select all that apply*)

For software verification and validation related activities in general:

	Yes, Open Source	Yes, Proprietary	Yes, In House	No	Do Not Know
Software Design Reviews	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐

For security-based software verification and validation activities:

	Yes, Open Source	Yes, Proprietary	Yes, In House	No	Do Not Know
Security Design Reviews	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐

Section 1: Which tools?

Question 11 (optional): Which tools are used?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Are the tools customised?

Question 12: Are these tools customised to your particular environment?

	Yes	No	Do Not Know
Software Design Reviews	○	○	○
Software Code Reviews	○	○	○
Software Testing	○	○	○
Security Design Reviews	○	○	○
Security Code Reviews	○	○	○
Security Testing	○	○	○
Penetration Testing	○	○	○

Section 1: Why are tools not utilised with these activities?

Question 13: Can you please elaborate on why tools are not used for the following activities? (*please select all that apply*)

	Lack of training / knowledge	Lack of budget	Lack of time	Lack of people	Lack of management support	Lack of motivation	Other
Software Design Reviews	☐	☐	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐	☐	☐
Security Design Reviews	☐	☐	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐	☐	☐

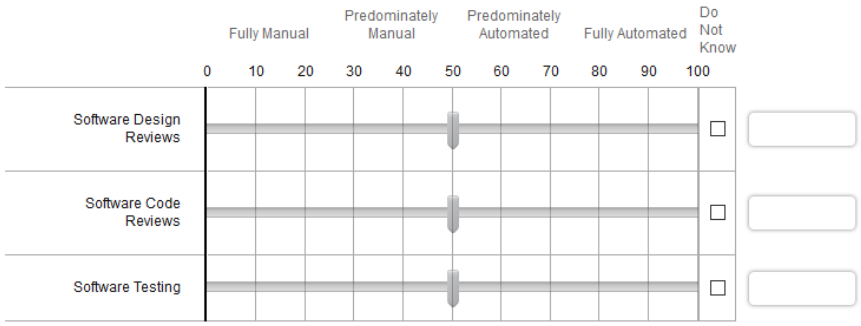
Question 14 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

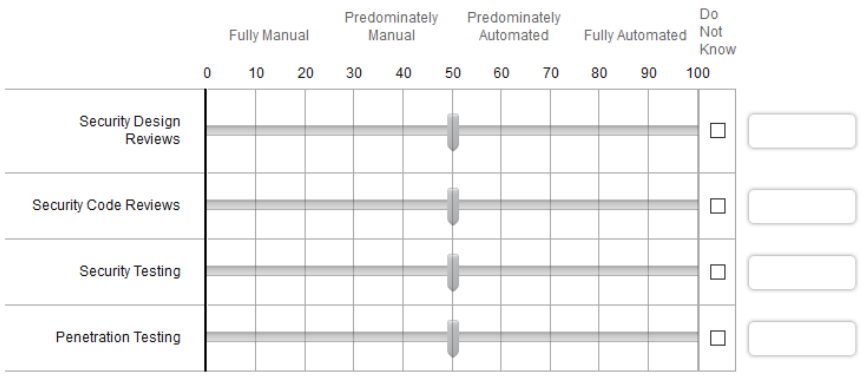
Section 1: How well is automation employed with these activities?

Question 15: As a percentage, approximately how much of the following activities are automated within your organisation?

For software verification and validation related activities in general:



For security-based software verification and validation activities:



Section 1: How do you communicate the results of these activities?

Question 16: How are the results from the following activities communicated? (*please select all that apply*)

For software verification and validation related activities in general:

	Formal Document	Email	Verbally	Other
Software Design Reviews	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐

For security-based software verification and validation activities:

	Formal Document	Email	Verbally	Other
Security Design Reviews	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐

Question 17 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Who do you communicate the results of these activities to?

Question 18: Who receives the results of the following activities within your organisation? (*please select all that apply*)

For software verification and validation related activities in general:

	Development	Test	Outsourced Engineers	Customer	Project Managers	Other
Software Design Reviews	☐	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐	☐

For security-based software verification and validation activities:

	Development	Test	Outsourced Engineers	Customer	Project Managers	Other
Security Design Reviews	☐	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐	☐

Question 19 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Do you have processes which cover these activities?

Question 20: Are there processes in place for the following activities?

For software verification and validation related activities in general:

	Yes	No	Do Not Know
Software Design Reviews	○	○	○
Software Code Reviews	○	○	○
Software Testing	○	○	○

For security-based software verification and validation activities:

	Yes	No	Do Not Know
Security Design Reviews	○	○	○
Security Code Reviews	○	○	○
Security Testing	○	○	○
Penetration Testing	○	○	○

Section 1: What are the origins of these processes?

Question 21: Are the processes in place based upon an existing standard?

	Yes, Partly	Yes, Fully	No	Do Not Know
Software Design Reviews	○	○	○	○
Software Code Reviews	○	○	○	○
Software Testing	○	○	○	○
Security Design Reviews	○	○	○	○
Security Code Reviews	○	○	○	○
Security Testing	○	○	○	○
Penetration Testing	○	○	○	○

Section 1: Which standards?

Question 22 (optional): Which standards are the processes based upon?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Have you received training in these activities?

Question 23: Regardless of whether the activities are carried out within your organisation, have you received training related to the following activities within your organisation?

For software verification and validation related activities in general:

	Yes	No
Software Design Reviews	○	○
Software Code Reviews	○	○
Software Testing	○	○

For security-based software verification and validation activities:

	Yes	No
Security Design Reviews	○	○
Security Code Reviews	○	○
Security Testing	○	○
Penetration Testing	○	○

Section 1: How was the training delivered for these activities?

Question 24: Can you please elaborate on the type of training you have received? (*please select all that apply*)

	Formal Training Course	Mentoring	Conference Attendance	Reading	Other
Software Design Reviews	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐
Security Design Reviews	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐

Question 25 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: How often are you trained in these activities?

Question 26: Can you please describe how often you have received training in the last two years?

For software verification and validation related activities in general:

	How many times was training received?	Approximately when was such training last provided?
Software Design Reviews		
Software Code Reviews		
Software Testing		

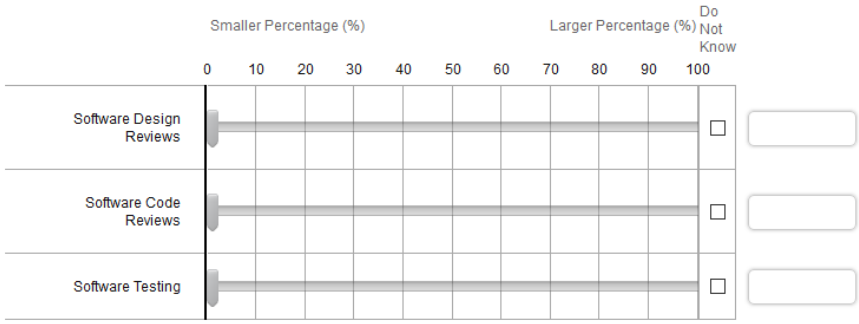
For security-based software verification and validation activities:

	How many times was training received?	Approximately when was such training last provided?
Security Design Reviews		
Security Code Reviews		
Security Testing		
Penetration Testing		

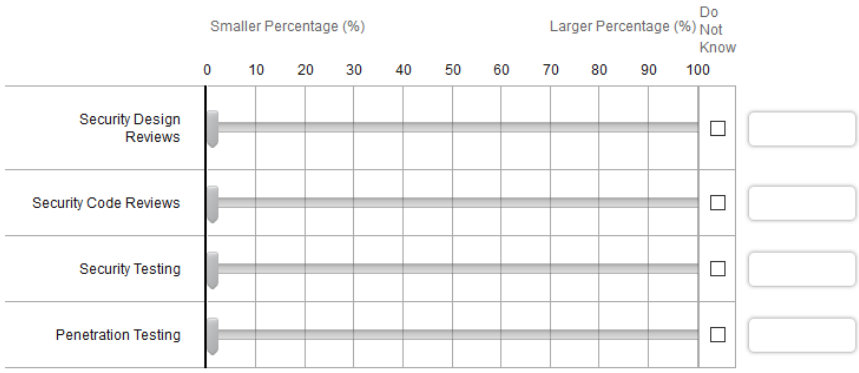
Section 1: How much is spent on these activities?

Question 27: As a percentage, approximately how much of a typical project's budget is spent on the following activities within your organisation?

For software verification and validation related activities in general:



For security-based software verification and validation activities:



Section 1: Should there be more investment in these activities?

Question 28: Regardless of whether the activities are carried out within your organisation, do you believe that there should be more investment in the following activities within your organisation?

For software verification and validation related activities in general:

	Yes	No
Software Design Reviews	○	○
Software Code Reviews	○	○
Software Testing	○	○

For security-based software verification and validation activities:

	Yes	No
Security Design Reviews	○	○
Security Code Reviews	○	○
Security Testing	○	○
Penetration Testing	○	○

Section 1: What resourcing constraints impact these activities?

Question 29: What resourcing constraints impact the following activities within your organisation? (*please select all that apply*)

For software verification and validation related activities in general:

	Lack of training / knowledge	Lack of budget	Lack of time	Lack of people	Lack of tool support	Other	N / A
Software Design Reviews	☐	☐	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐	☐	☐

For security-based software verification and validation activities:

	Lack of training / knowledge	Lack of budget	Lack of time	Lack of people	Lack of tool support	Other	N / A
Security Design Reviews	☐	☐	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐	☐	☐

Question 30 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: What social factors impact these activities?

Question 31: What social factors impact the following activities within your organisation? (*please select all that apply*)

For software verification and validation related activities in general:

	Poor team work	Poor communication	Lack of motivation	Lack of management support	Activity viewed as dispensable	Other	N / A
Software Design Reviews	☐	☐	☐	☐	☐	☐	☐
Software Code Reviews	☐	☐	☐	☐	☐	☐	☐
Software Testing	☐	☐	☐	☐	☐	☐	☐

For security-based software verification and validation activities:

	Poor team work	Poor communication	Lack of motivation	Lack of management support	Activity viewed as dispensable	Other	N / A
Security Design Reviews	☐	☐	☐	☐	☐	☐	☐
Security Code Reviews	☐	☐	☐	☐	☐	☐	☐
Security Testing	☐	☐	☐	☐	☐	☐	☐
Penetration Testing	☐	☐	☐	☐	☐	☐	☐

Question 32 (optional): Can you please elaborate on your selection of "other" for the following activities?

Software Design Reviews	
Software Code Reviews	
Software Testing	
Security Design Reviews	
Security Code Reviews	
Security Testing	
Penetration Testing	

Section 1: Do you believe these activities are carried out well?

Question 33: Do you believe that the following activities are carried out well within your organisation?

For software verification and validation related activities in general:

	Yes	No	Do Not Know
Software Design Reviews	○	○	○
Software Code Reviews	○	○	○
Software Testing	○	○	○

For security-based software verification and validation activities:

	Yes	No	Do Not Know
Security Design Reviews	○	○	○
Security Code Reviews	○	○	○
Security Testing	○	○	○
Penetration Testing	○	○	○

Section 1: How do you perceive the importance of these activities?

Question 34: How would you rank the importance of the following activities? (*with 1 being the most important and 7 the least important - left-click and hold your mouse button on any activity, then drag it up or down to change its rank*)

Software Design Reviews
Software Code Reviews
Software Testing
Security Design Reviews
Security Code Reviews
Security Testing
Penetration Testing

Question 35 (optional): Please feel free to elaborate on your choices above.

H.3 Section Two: Information Security Culture

Section 2: Information Security Culture

In this section we aim to understand what type of information security culture exists within your organisation.

Please click the "NEXT" button to begin this section.

Section 2: Is there an information security policy?

Question 1: Does your organisation have an information security policy?

- ○ Yes
- ○ No
- ○ Do Not Know

Section 2: Is this information security policy understood and followed?

Question 2: Do you understand the information security policy?

- ○ Yes
- ○ No
- ○ To Some Extent

Question 3: Do you believe that your fellow employees are aware of the information security policy's existence?

- ○ Yes
- ○ No
- ○ Do Not Know

Question 4: Do you believe that your fellow employees understand the information security policy?

- ○ Yes
- ○ No
- ○ To Some Extent
- ○ Do Not Know

Question 5: Do you believe that your organisation's senior management follows the information security policy?

- ○ Yes
- ○ No
- ○ To Some Extent
- ○ Do Not Know

Section 2: Are you notified of updates to the information security policy?

Question 6: When updates are made to the information security policy are you notified of these updates?

- ○ Yes
- ○ No

Section 2: How are you notified of updates to the information security policy?

Question 7: How are you notified? (*please select all that apply*)

- ☐ Email
- ☐ Verbally
- ☐ Leaflets
- ☐ Flyers / Posters
- ☐ Other:

Section 2: Have you received any information security training?

Question 8: Did you receive information security training during your induction?

- ○ Yes
- ○ No

Question 9: Have you received any subsequent information security training?

- ○ Yes
- ○ No
- ○ Do Not Recall

Section 2: How often do you receive information security training?

Question 10: On average, how often do you receive information security training per year?

- ○ 1 time
- ○ 2 times
- ○ 3 - 5 times
- ○ 6 - 12 times
- ○ Over 12 times

Section 2: Are you and your fellow employees security aware?

Question 11: Do you consider yourself to be security aware?

- ○ Yes
- ○ No
- ○ To Some Extent

Question 12: Do you consider your fellow employees to be security aware?

- ○ Yes
- ○ No
- ○ To Some Extent
- ○ Do Not Know

Section 2: Have you attempted to improve security awareness?

Question 13: Have you attempted to raise the level of security awareness within your organisation?

- ○ Yes
- ○ No

Question 14: Apart from you has anyone else within your organisation attempted to raise the level of security awareness?

- ○ Yes
- ○ No
- ○ Do Not Know

Section 2: What have you done to raise security awareness?

Question 15 (optional): What have you done, or do, to raise security awareness within your organisation?

Question 16 (optional): What motivated you to do this?

Question 17: Was this effective in raising security awareness?

- ○ Yes
- ○ No
- ○ To Some Extent
- ○ Do Not Know

Question 18 (optional): Why was this effective?

Question 19 (optional): Why was this ineffective?

Question 20 (optional): Why was this not as effective as it could have been?

Section 2: What have others done to raise security awareness?

Question 21 (optional): What have others done, or do, to raise security awareness within your organisation?

Question 22: Was this effective in raising security awareness?

- ○ Yes
- ○ No
- ○ To Some Extent
- ○ Do Not Know

Question 23 (optional): Why was this effective?

Question 24 (optional): Why was this ineffective?

Question 25 (optional): Why was this not as effective as it could have been?

Section 2: Who is responsible for information security?

Question 26: Who is responsible for information security within your organisation?

- ○ A Single Individual
- ○ A Group of Individuals
- ○ Do Not Know
- ○ No One

Section 2: How is this individual placed within the organisation?

Question 27: Is this individual dedicated to this function or do they hold other roles within the organisation?

- ○ Dedicated
- ○ Other Roles
- ○ Do Not Know

Question 28: Do they report to senior management?

- ○ Yes
- ○ No
- ○ They Are Senior Management
- ○ Do Not Know

Section 2: How is this group placed within the organisation?

Question 29: How many people hold this responsibility within your organisation?

- ○ Between 2 - 5
- ○ Between 6 - 10
- ○ Over 10
- ○ Other:

Question 30: Is this group of individuals dedicated to this function or do they hold other roles within the organisation?

- ○ Dedicated
- ○ Other Roles
- ○ Do Not Know

Question 31: Is there at least one member of this group who reports to senior management?

- ○ Yes
- ○ No
- ○ They Are Senior Management
- ○ Do Not Know

Section 2: Are you aware of your information security responsibilities?

Question 32: Are you aware of your responsibilities within the organisation to help maintain security?

- ○ Yes
- ○ No
- ○ To Some Extent

Question 33: Does your contract of employment incorporate these responsibilities?

- ○ Yes
- ○ No
- ○ To Some Extent
- ○ Do Not Know

Question 34: Are you aware of your organisation's responsibilities to help maintain security?

- ○ Yes
- ○ No
- ○ To Some Extent

Section 2: Does your organisation have an information security culture?

Question 35: Do you believe an information security culture exists within your organisation?

- ○ Yes
- ○ No
- ○ Do Not Know

Section 2: How strong is the information security culture?

Question 36: Do you consider the security culture to be weak or strong?

Very Weak	Weak	Strong	Very Strong
○	○	○	○

H.4 Section Three: Demographics

Section 3: Demographics

In this section we ask a small number of demographic type questions.

Please click the "NEXT" button to begin this section.

Section 3: Information on your organisation

Question 1: What type of organisation do you currently work for?

- ○ Very Small (< 10 employees)
- ○ Small (< 50 employees)
- ○ Medium (< 250 employees)
- ○ Large (250 employees and over)
- ○ Specify number of employees if known:

Question 2: What is the industry sector at which your organisation's software is mainly targetted?

Question 3: How old is your current organisation?

- ○ Less than 1 year
- ○ 1 - 2 years
- ○ 3 - 5 years
- ○ 6 - 10 years
- ○ 11 - 20 years
- ○ Over 20 years
- ○ Specify age if known:

Question 4: What software development methodologies do you use within your organisation? (*please select all that apply*)

- ☐ Agile
- ☐ Test-Driven Development
- ☐ V-Model
- ☐ Waterfall
- ☐ Other:

Section 3: Information about your role

Question 5: Which of the following categories best describes your current occupation?

- ○ Developer
- ○ Test Engineer
- ○ Other:

Question 6: Which of the following levels best reflects your position within your organisation?

- ○ Junior
- ○ Senior
- ○ Lead
- ○ Manager
- ○ Senior Manager

Question 7: How long have you worked for your current organisation?

- ○ Less than 1 year
- ○ 1 - 2 years
- ○ 3 - 5 years
- ○ 6 - 10 years
- ○ 11 - 20 years
- ○ Over 20 years
- ○ Specify time if known:

Question 8: Which country is your primary location?

Section 3: Information about your previous experience

Question 9: What types of role have you previously carried out? (*please select all that apply*)

- ☐ Developer
- ☐ Test Engineer
- ☐ Other:
- ☐ N / A

Question 10: How long have you worked within the software industry?

- ○ Less than 1 year
- ○ 1 - 2 years
- ○ 3 - 5 years
- ○ 6 - 10 years
- ○ 11 - 20 years
- ○ Over 20 years
- ○ Specify time if known:

Question 11: What types of organisations have you previously worked for? (*please select all that apply*)

- ☐ Very Small (< 10 employees)
- ☐ Small (< 50 employees)
- ☐ Medium (< 250 employees)
- ☐ Large (250 employees and over)
- ☐ N / A

Section 3: Contact information

Question 12: Would you like to receive a copy of the results when made available? (*this would enable you to identify your organisation's position amongst similar organisations*)

- ○ Yes
- ○ No

Question 13: Would you be willing to participate in a short follow-up interview?

○ Yes

○ No

Question 14: Please enter your email address. (*your email address will only be used to contact you to arrange an interview or to send you the results of the survey*)

Question 15: How would you like to be contacted? (*please select all that apply*)

☐ Face to face discussion

☐ Video conferencing

☐ Telephone

☐ Instant Messaging

☐ Email

H.5 End Page

Thank you very much for your time

If you have any additional comments, either concerning your answers, or the survey itself, please indicate these below:

Appendix I

Survey Respondent Information

This appendix provides context to the respondents involved with the first phase of data collection and analysis. Specifically, providing information on their organisations (Table I.1), roles (Table I.2) and experience (Table I.3). The findings of the first phase of data collection are provided within Chapter 5.

Table I.1: Survey respondent organisation information

ID	Reference	Organisation		Industry Sector Targetted	Software Development Methodology				
		Size	Age (Years)		Agile	TDD	V-Model	Waterfall	Other
1	[O:KD]	Medium	11 - 20	Other	✓	✓	×	×	-
2	[O:CS]	Medium	11 - 20	Information and communication	✓	×	×	✓	-
3	[O:DS]	Small	3 - 5	Information and communication	×	×	✓	×	-
4	[O:SP]	Medium	11 - 20	Administrative and support service activities	✓	×	×	×	-
5	[O:DF]	Small	20+	Education	✓	×	×	×	Customer driven
6	[O:AV]	Medium	6 - 10	Information and communication	✓	×	✓	×	-
7	[O:PB]	Medium	11 - 20	Information and communication	✓	×	×	×	-
8	[O:TN]	Medium	3 - 5	Information and communication	✓	✓	✓	×	-
9	[O:RV]	Medium	11 - 20	Other	×	✓	×	×	-
10	[O:MF]	Medium	20+	Administrative and support service activities	✓	×	×	✓	-
11	[O:WB]	Medium	20+	Human health and social work activities	✓	×	×	✓	-
12	[O:MC]	Medium	20+	Information and communication	×	×	✓	×	-
13	[O:CB]	Medium	11 - 20	Other service activities	×	×	×	✓	-
14	[O:FR]	Small	6 - 10	Financial and insurance activities	✓	×	×	×	-
15	[O:WS]	Medium	11 - 20	Professional, scientific and technical activities	✓	✓	×	×	-
16	[O:WK]	Medium	11 - 20	Information and communication	✓	×	✓	✓	-
17	[O:CC]	Small	20+	Human health and social work activities	✓	×	✓	✓	-
18	[O:BN]	Medium	6 - 10	Other	✓	✓	×	×	-
19	[O:AD]	Small	20+	Construction	✓	×	×	✓	-
20	[O:SS]	Small	20+	Information and communication	✓	✓	×	×	-

Table I.1: Survey respondent organisation information (continued)

ID	Reference	Organisation		Industry Sector Targetted	Software Development Methodology				
		Size	Age (Years)		Agile	TDD	V-Model	Waterfall	Other
21	[O:CR]	Medium	11 - 20	Information and communication	✓	×	✓	✓	-
22	[O:EB]	Medium	20+	Financial and insurance activities	✓	×	✓	✓	-
23	[O:CN]	Medium	11 - 20	Transportation and storage	✓	✓	×	×	-
24	[O:OS]	Medium	20+	Human health and social work activities	✓	✓	×	×	-
25	[O:WC]	Medium	1 - 2	Financial and insurance activities	✓	✓	✓	✓	-
26	[O:CD]	Medium	6 - 10	Other	✓	✓	×	×	-
27	[O:CH]	Medium	11 - 20	Other	✓	✓	×	×	-
28	[O:DN]	Medium	11 - 20	Administrative and support service activities	✓	×	✓	×	-
29	[O:VS]	Medium	11 - 20	Other	✓	×	×	×	-
30	[O:CT]	Small	6 - 10	Electricity, gas, steam and air conditioning supply	✓	✓	×	×	-
31	[O:SV]	Small	6 - 10	Information and communication	✓	×	×	×	-
32	[O:AG]	Small	6 - 10	Other	✓	✓	×	×	-
33	[O:US]	Medium	11 - 20	Professional, scientific and technical activities	×	✓	✓	✓	Not fixed
34	[O:UB]	Medium	6 - 10	Information and communication	✓	✓	✓	✓	-
35	[O:SC]	Medium	6 - 10	Information and communication	✓	×	×	×	-
36	[O:PM]	Small	11 - 20	Education	✓	×	×	✓	-
37	[O:RW]	Medium	11 - 20	Other	✓	×	×	×	-

Table I.2: Survey respondent role information

ID	Reference	Title	Position	Role	Tenure (Years)
1	[O:KD-I:MF]	Head of Engineering	Senior Manager	Other: Development Head	11 - 20
2	[O:CS-I:GB]	Chief Technology Officer	Senior Manager	Other: CTO	3 - 5
3	[O:DS-I:SW]	Chief Technology Officer	Senior Manager	Other: CTO	3 - 5
4	[O:SP-I:WS]	Product Development Director	Manager	Developer	20+
5	[O:DF-I:JR]	Senior Test Systems Engineer	Senior	Test Engineer	3 - 5
6	[O:AV-I:JG]	Head of Quality Assurance	Manager	Test Engineer	3 - 5
7	[O:PB-I:SK]	Senior Test Lead	Lead	Test Engineer	1 - 2
8	[O:TN-I:AE]	Security and Crypto Specialist	Senior	Developer	<1
9	[O:RV-I:LH]	Customer Support Manager	Manager	Other: Technical support	3 - 5
10	[O:MF-I:TE]	Product Development Manager	Senior Manager	Other: IT Director	1 - 2
11	[O:WB-I:LN]	Test Team Lead	Lead	Test Engineer	3 - 5
12	[O:MC-I:AP]	Test Manager	Manager	Test Engineer	11 - 20
13	[O:CB-I:PT]	Quality Assurance Manager	Manager	Test Engineer	3 - 5
14	[O:FR-I:DK]	Lead QA Analyst	Lead	Test Engineer	1 - 2
15	[O:WS-I:WE]	QA Director	Senior Manager	Other: Test Management	6 - 10
16	[O:WK-I:RR]	Head of Testing	Senior Manager	Other: Head of Testing	<1
17	[O:CC-I:PC]	Quality Assurance Manager	Manager	Test Engineer	6 - 10
18	[O:BN-I:AC]	Development Manager	Senior Manager	Other: Development Director	3 - 5
19	[O:AD-I:MD]	Head of Development	Senior Manager	Developer	11 - 20
20	[O:SS-I:RD]	Product Development Manager	Senior Manager	Other: Product Manager	3 - 5
21	[O:CR-I:MC]	Senior QA Manager	Senior Manager	Test Engineer	6 - 10
22	[O:EB-I:MH]	Development Manager	Manager	Developer	<1
23	[O:CN-I:DI]	Automation Test Team Leader	Lead	Test Engineer	1 - 2
24	[O:OS-I:ST]	Head of Applications Development	Senior Manager	Other: Head of Development	11 - 20
25	[O:WC-I:AC]	Lead QA	Lead	Test Engineer	1 - 2
26	[O:CD-I:KS]	QA Test Lead	Lead	Test Engineer	3 - 5
27	[O:CH-I:ZG]	Quality Assurance Team Lead	Junior	Test Engineer	3 - 5

Table I.2: Survey respondent role information (continued)

ID	Reference	Title	Position	Role	Tenure (Years)
28	[O:DN-I:SA]	QA Analyst	Lead	Test Engineer	1 - 2
29	[O:VS-I:GH]	Scrum Master and Lead QA Engineer	Lead	Test Engineer	<1
30	[O:CT-I:DS]	Software Tester	Senior	Test Engineer	<1
31	[O:SV-I:TD]	Test Engineer	Lead	Test Engineer	1 - 2
32	[O:AG-I:DS]	Lead Integration Engineer	Senior	Test Engineer	6 - 10
33	[O:US-I:GJ]	Test and Support Manager	Senior	Test Engineer	3 - 5
34	[O:UB-I:SP]	Lead Software Test Engineer	Lead	Test Engineer	<1
35	[O:SC-I:CB]	Senior Test Development Engineer	Lead	Test Engineer	3 - 5
36	[O:PM-I:AK]	Lead Test Engineer	Lead	Test Engineer	<1
37	[O:RW-I:RL]	Test Manager	Senior	Test Engineer	1 - 2

Table I.3: Survey respondent experience

ID	Reference	Experience (Years)	Previous Roles Held				Types of Organisations Worked For				
			Developer	Test Engineer	Other	N/A	Very Small	Small	Medium	Large	N/A
1	[O:KD-I:MF]	20+	✓	×	-	×	✓	✓	×	×	×
2	[O:CS-I:GB]	20+	✓	×	-	×	×	✓	✓	✓	×
3	[O:DS-I:SW]	20+	✓	×	-	×	×	×	×	✓	×
4	[O:SP-I:WS]	20+	✓	×	-	×	×	×	×	✓	×
5	[O:DF-I:JR]	11 - 20	×	✓	-	×	✓	✓	✓	✓	×
6	[O:AV-I:JG]	11 - 20	✓	✓	-	×	×	✓	✓	✓	×
7	[O:PB-I:SK]	6 - 10	×	✓	-	×	×	×	×	✓	×
8	[O:TN-I:AE]	6 - 10	✓	×	-	×	✓	×	×	✓	×
9	[O:RV-I:LH]	11 - 20	×	×	-	✓	×	✓	✓	✓	×
10	[O:MF-I:TE]	20+	×	✓	-	×	×	×	×	✓	×
11	[O:WB-I:LN]	6 - 10	×	✓	-	×	×	×	✓	✓	×
12	[O:MC-I:AP]	11 - 20	×	✓	-	×	×	×	×	✓	×
13	[O:CB-I:PT]	11 - 20	×	✓	-	×	×	×	×	✓	×
14	[O:FR-I:DK]	6 - 10	×	✓	-	×	×	✓	✓	✓	×
15	[O:WS-I:WE]	11 - 20	×	✓	-	×	×	×	✓	✓	×
16	[O:WK-I:RR]	11 - 20	×	✓	Test Manager	×	×	×	×	✓	×
17	[O:CC-I:PC]	20+	×	✓	-	×	×	×	✓	✓	×
18	[O:BN-I:AC]	11 - 20	✓	×	-	×	×	✓	×	×	×
19	[O:AD-I:MD]	20+	✓	×	-	×	×	✓	×	×	×
20	[O:SS-I:RD]	11 - 20	×	✓	-	×	✓	✓	×	✓	×
21	[O:CR-I:MC]	20+	×	✓	DBA	×	×	✓	✓	✓	×
22	[O:EB-I:MH]	20+	✓	✓	Project Manager	×	×	×	✓	×	×
23	[O:CN-I:DI]	6 - 10	×	×	-	✓	×	×	×	✓	×
24	[O:OS-I:ST]	20+	✓	×	Team Manager	×	✓	×	✓	✓	×
25	[O:WC-I:AC]	6 - 10	×	✓	-	×	×	×	×	✓	×

Table I.3: Survey respondent experience (continued)

ID	Reference	Experience (Years)	Previous Roles Held				Types of Organisations Worked For				
			Developer	Test Engineer	Other	N/A	Very Small	Small	Medium	Large	N/A
26	[O:CD-I:KS]	6 - 10	×	✓	-	×	✓	×	×	×	×
27	[O:CH-I:ZG]	3 - 5	×	×	Service Desk Consultant	×	×	×	×	✓	×
28	[O:DN-I:SA]	6 - 10	×	✓	-	×	×	✓	✓	✓	×
29	[O:VS-I:GH]	6 - 10	×	✓	-	×	×	×	×	✓	×
30	[O:CT-I:DS]	11 - 20	×	✓	-	×	×	✓	✓	✓	×
31	[O:SV-I:TD]	6 - 10	×	✓	-	×	×	✓	×	✓	×
32	[O:AG-I:DS]	11 - 20	×	✓	-	×	×	✓	✓	×	×
33	[O:US-I:GJ]	20+	✓	✓	Support Manager	×	×	✓	✓	✓	×
34	[O:UB-I:SP]	6 - 10	×	✓	Support	×	×	×	✓	✓	×
35	[O:SC-I:CB]	3 - 5	×	✓	-	×	×	×	×	×	✓
36	[O:PM-I:AK]	11 - 20	×	✓	-	×	×	×	×	✓	×
37	[O:RW-I:RL]	6 - 10	×	✓	Multiple outside of the SDLC	×	✓	✓	✓	✓	×

Appendix J

Interview Guide Pre-Testing

This appendix details the interview guide pre-testing performed. The pre-testing sessions took the following format:

- We spent approximately 40 minutes conducting the interview during the first session. A further 20 minutes were spent discussing the questions and reflecting upon the interview.
- Almost 70 minutes were spent on the interview during the second session, and approximately 50 minutes on subsequent discussion and reflection.

Whilst it was acknowledged, during both sessions, that the questions flowed well, it became apparent, during the first session, that the interviewee appreciated the elaboration given on a number of questions. In particular, it was suggested that we define V&V, and thus the various activities encompassed, at the start of the interview. This feedback resulted in the expansion of the interview introduction to include some additional, relevant definitions. The only change resulting from the second session was that we should explicitly seek clarification on whether the security testing being performed within an organisation is "*[m]ore black-box or white-box*". This was a sensible suggestion and demonstrates the importance of involving engaged subject matter experts and, more generally, for pre-testing an interview guide (see Section 6.1). Table J.1 details the interviewees involved in the interview guide pre-testing.

Table J.1: Interview guide pre-testing: interviewee profile and relevance

ID	Type of Respondent	Technical Relevance	Organisational Relevance				Language (English)	Experience (Years)
			Very Small	Small	Medium	Large		
1	Subject matter expert	V&V practitioner	✓	×	×	✓	First	6 - 10
2	Subject matter expert	V&V practitioner	×	✓	✓	✓	Second	6 - 10

Appendix K

Interview Guide

The interview guide used for the second phase of data collection (see Chapter 6) is presented within this appendix, with the pre-testing performed captured within Section 6.1. The guide was developed on the basis of the survey (including the question themes presented in Appendix D) and through analysis of the data obtained during the first phase of data collection (see Chapter 5). In addition, before conducting an interview, the following items were discussed with the interviewees:

- A high-level overview of the research was given (reiterating much of what was captured in the initial communication, see Appendix G).
- A definition of terms was provided to ensure a common understanding (encompassing both the V&V activities and information security culture).
- The interview format was detailed.
- That ethical approval for conducting the study had been obtained was stated.
- Approval for recording the interview was sought.

The interview guide comprises two sections: an initial set of demographic questions (Section K.1) and the main questions (Section K.2).

K.1 Demographic Questions

Question 1: Which of the following categories best describes your current occupation?
- Developer
- Test Engineer
- Other: <specify>

Question 2: Which of the following levels best reflects your position within your organisation?
- Junior
- Senior
- Lead
- Manager
- Senior Manager

Question 3: How long have you worked for your current organisation?
- Less than 1 year
- 1 - 2 years
- 3 - 5 years
- 6 - 10 years
- 11 - 20 years
- Over 20 years
- Specify time if known: <specify>

Question 4: What types of role have you previously carried out?
- Developer
- Test Engineer
- Other: <specify>
- N/A

Question 5: How long have you worked within the software industry?
- Less than 1 year
- 1 - 2 years
- 3 - 5 years
- 6 - 10 years
- 11 - 20 years
- Over 20 years
- Specify time if known: <specify>

Question 6: What types of organisations have you previously worked for?
- Very Small (< 10 employees)
- Small (< 50 employees)
- Medium (< 250 employees)
- Large (250 employees or more)
- N/A

K.2 Interview Questions

Structure of Engineering Team

Question 1: What is the structure of the engineering team?

- What are the distinct groups of engineers by role/discipline?
- How do these groups report into the "head of engineering"?

Question 2: Does the "head of engineering" influence the engineering team?

- Is the "head of engineering" more a developer or a tester?
- Does this influence the level of software V&V being performed and the support for the activities?
- Are they security focused?
- Does this influence the level of focus on security activities, including security V&V, and the support for the activities?

Question 3: Do any of the software or security V&V activities practiced within your organisation have a recognised champion?

- If "yes": Which activities and where do they sit?
- If "no": Why not?
- Is any one dedicated to performing security V&V within your organisation?

Developers and Testers

Question 4: Are developers happy to perform test-related activities?

- If "yes": Why?
- If "no": Why not?
- Which test-related activities are developers most likely to want to perform: software or security and why?

Question 5: What is the relationship like between developers and testers?

- If positive: How was this relationship cultivated?
- If negative: Why is this and how does this exhibit itself?
- If positive or negative: How does this relationship influence the software and security V&V activities practiced within your organisation?
- What is the communication like between developers and testers?
  - How do they feel when testers raise bugs against their code?
- Who has more power and how does this exhibit itself?

Question 6: Are developers and testers perceived as being equal within your organisation?

- If "yes": Why?
- If "no": Why not?
- If "no" and respondent is a test engineer: Does this bother you?

Role Conflict

Question 7: Has role conflict influenced the software and security V&V activities practiced within your organisation?

- If "yes": How?
- If "no": Why?
- As an SME, do you think you are impacted by multiple-hat syndrome?

Stakeholders Outside of Engineering

Question 8: What is the relationship like between engineering and the rest of the organisation?

- Do those outside of engineering see the importance in the following activities:
  - Software V&V?
  - Software security?
  - Security V&V?
- Do those outside of engineering respect the engineering team?
  - If "yes": Why?
  - If "no": Why not?
- Who do they respect the most within engineering and why?

Question 9: Who are the stakeholders outside of engineering?

- Do they have more power than the engineering team?

Security V&V and Software Security

Question 10: At what point in the software development lifecycle is software V&V and security V&V applied?

Question 11: Is your security testing more functional or risk-based in nature? More black-box or white-box?

Question 12: How are the software and security V&V activities supported within your organisation?

Question 13: How is the communication surrounding the software V&V and security V&V activities?

Question 14: What are the incentives and the motivation within your organisation for achieving a secure product?

- Internal or external to your organisation?
- Are people motivated to perform security V&V? Why is this?

Question 15: Does your organisation follow any secure development process?

Impact of Release Deadlines and Vulnerabilities

Question 16: Who determines when the product is ready to be released?

Question 17: Do software and security V&V activities get squeezed as release deadlines loom?

- Who enforces/applies this pressure?
- Which activities are most likely to get squeezed?
- How do you feel when facing this pressure?

Question 18: Would finding a vulnerability in your product hold up a release?

- If "yes": Why?
- If "no": Why not?
- How are vulnerabilities tagged in your bug tracking system?
- Are they given more or less priority than other bugs?
- Who reports the majority of the vulnerabilities?
- What activities find the majority of the vulnerabilities?

What Impacts Security V&V

Question 19: What impacts the software V&V and security V&V activities within your organisation?

- What could be done to improve security V&V within your organisation? More money? Training?
- Is support available to improve and do you think it would be successful?
- Which is the most important point to improve upon?

Question 20: Is security perceived as "just another" software quality attribute within your organisation?

People Excluded from Security V&V

Question 21: Is there anyone within your organisation who should be involved with security V&V and is not?

- If "yes": Who and why?

Use of Outsourcing

Question 22: Should software and security V&V activities be outsourced?

- Which activities should be outsourced and why?
- Which activities should not be outsourced and why?

Software Process Improvement

Question 23: Is your organisation currently undergoing any form of software process improvement?

- For:
  - Software development activities?
  - Software V&V activities?
  - Security V&V activities?
- If undertaking some form of software process improvement:
  - What form does this take?
  - Who is driving this improvement?
  - Do you think it will be successful?
    - If "yes": Why?
    - If "no": Why not?
- If not undertaking some form of software process improvement:
  - Why?
  - Has your organisation previously invested in any form of software process improvement?

Software Engineering Maturity

Question 24: How mature is your organisation's software engineering practice?

- For:
  - Software development activities?
  - Software V&V activities?
  - Security V&V activities?
- If "Software V&V" and "Security V&V" differ: Why is this?
- Do you think your security V&V practice can be improved separate to improving your software V&V practice?

Question 25: Since the time of the survey, has the situation changed within your organisation regarding software V&V and/or security V&V? If so, how?

- Has your view of the importance of the various software V&V and security V&V activities changed? Is this view reflected by your organisation?

Information Security Culture and Security V&V

Question 26: People typically define an information security culture as when security has become "a natural aspect of all the organization's employees' daily activities". Do you believe this is the case within your organisation?

- How has this influenced the security V&V being performed within your organisation?
- Do you think improving your organisation's information security culture would help improve your security V&V practice?
- Which do you think is easier to improve, your organisation's information security culture or security V&V practice? Why?

Appendix L

Interview Respondent Information

This appendix provides context to the interviewees involved with the second phase of data collection and analysis (Table L.1). Whilst we acknowledge that "qualitative interviews cannot study a very large or random sample of people, due to the large amount of time and effort involved and limitation of access" [Qu and Dumay, 2011, p. 244], we note that "the number of interviewed experts can be rather low, as long as they were selected carefully" [Overhage et al., 2011]. We believe that our interviewees were carefully selected, thereby continuing our theme of selecting experienced, senior employees (see Section 5.4). Similarly, others have also identified that due to the inherent richness afforded by qualitative data, a small number of interviews can be sufficient e.g. [Ammenwerth, 2010, Braun and Clarke, 2013]. This is particularly the case when interviewing a small number of well-selected individuals [Cleary et al., 2014] from a relatively homogeneous population [Boddy, 2016] (since, as captured by Boddy: qualitative research often aims to develop a depth of understanding rather than a breadth). Notably, Guest et al. posit that "[f]or most research enterprises, however, in which the aim is to understand common perceptions and experiences among a group of relatively homogeneous individuals, twelve interviews should suffice" [Guest et al., 2006, p. 79]. The findings of the second phase of data collection are provided within Chapter 6.

Table L.1: Interview respondent information

ID	Reference	Title	Organisation		Survey Participant	Interview Mechanism
			Size	Age (Years)		
1	[O:RW-I:RL]	Test Manager	Medium	11 - 20	✓	Telephone
2	[O:SS-I:RD]	Product Development Manager	Small	20+	✓	Telephone
3	[O:CT-I:DS]	Software Tester	Small	6 - 10	✓	Face-to-face
4	[O:CD-I:KS]	QA Test Lead	Medium	6 -10	✓	Email
5	[O:KD-I:MF]	Head of Engineering	Medium	11 - 20	✓	Email
6	[O:VS-I:GH]	Scrum Master and Lead QA Engineer	Medium	11 - 20	✓	Email
7	[O:AG-I:DS]	Lead Integration Engineer	Small	6 - 10	✓	Face-to-face and email
8	[O:RV-I:SK]	Lead Software Test Engineer	Medium	11 - 20	×	Face-to-face
9	[O:RV-I:ML]	Lead Software Test Engineer	Medium	11 - 20	×	Face-to-face
10	[O:RV-I:BM]	Security Test Engineer	Medium	11 - 20	×	Face-to-face
11	[O:SP-I:WS]	Product Development Director	Medium	11 - 20	✓	Email
12	[O:DF-I:JR]	Senior Test Engineer	Small	20+	✓	Email

Appendix M

Software Security and Software V&V Maturity Models

This appendix details the software security maturity models (Section M.1), and the software V&V maturity models (Section M.2), used to devise, and support, the development of the questions used within the survey instrument (Appendix H) and the interview guide (Appendix K).

M.1 Software Security Maturity Models

Table M.1 details the software security maturity models referenced in Section 4.5.1.

Table M.1: Software security maturity models

Model	Description
Building Security In Maturity Model (BSIMM) [McGraw et al., 2016]	Allows an organisation to assess its maturity across a series of software security practices. Evolved from studying software security initiatives within 95 organisations, the model describes 113 distinct activities, spread across four domains, each containing three software security practices. The pertinent practices are: architecture analysis (including design reviews), code review, security testing and penetration testing. These activities encompass our definition of security V&V (see Section 1.3.5).

Table M.1: Software security maturity models (continued)

Model	Description
Software Assurance Maturity Model (SAMM) [OWASP, 2017]	Enables organisations to evaluate, and measure, their existing software security practices and can be used by both SMEs and large organisations. The framework contains four business functions, with each containing three software security practices. The verification business function encompasses "the processes and activities related to how an organization checks and tests artifacts produced throughout software development", which: "typically includes quality assurance work such as testing, but it can also include other review and evaluation activities". This definition encapsulates our focus on security V&V, specifically, test and review-based activities.

M.2 Software V&V Maturity Models

Table M.2 details the software V&V maturity models referenced in Section 4.5.2.

Table M.2: Software V&V maturity models

Model	Description
Test Process Improvement (TPI) Model [Koomen and Pol, 1999]	A means of assessing software test maturity within an organisation. The model contains 20 different areas of focus, including: test strategy, moment of involvement, test automation, commitment and motivation, communication and training. Each area of focus has a number of levels, comprising a number of checkpoints, which need to be satisfied for an organisation to meet a particular maturity level. TPI has been utilised in a number of studies e.g. [Koomen, 2002, Grindal et al., 2006a, Grindal et al., 2006b, Park et al., 2008].
Business Driven Test Process Improvement (BDTPI) Model [de Vries et al., 2009]	An updated version of the TPI model, including changes to some of the key areas of focus (with some areas being removed and some new areas being introduced).

Table M.2: Software V&V maturity models (continued)

Model	Description
Test Maturity Model integration (TMMi) [TMMi Foundation, 2018]	Created as a complementary model to CMMI (due to the "limited attention" given to software testing) to help address issues deemed important to test managers, test engineers and software quality professionals [TMMi Foundation, 2018, p. 6]. Five maturity levels are defined: "Initial", "Managed", "Defined", "Measured" and "Optimization". Each contains a set of process areas, comprising a set of goals and practices which, when met, allow an organisation to be rated at a particular level of maturity. TMMi has been adapted as a framework for assessing software test maturity within SMEs [Araújo et al., 2013] (although employed within four SMEs, the developed questionnaire contained 261 questions which, unsurprisingly, was considered an issue by respondents). TMMi has also formed the basis of other studies e.g. [Camargo et al., 2013, Camargo et al., 2015]; with one questionnaire (constructed in line with TMMi and providing a view on organisational test maturity) containing just 32 questions [Experimentus Ltd, 2012, Experimentus Ltd, 2020].
Inspection Capability Maturity Model (ICMM) [Kollanus, 2011]	A model for analysing an organisation's software review practices. Kollanus notes the similarity with both the CMM and CMMI models, however, rather than focusing on the entirety of the software development process, ICMM focuses solely upon software review practices. The model contains five maturity levels ("Initial" through to "Optimizing"), with each level including a series of key process areas and requirements which must be satisfied to meet that particular maturity level.