Oxford Review of Economic Policy, Volume 23, Number 4, 2007, pp.605–619

Why develop open-source software? The role of non-pecuniary benefits, monetary rewards, and open-source licence type

Robert M. Sauer*

Abstract A review of the basic theory of optimal open-source software contributions points to three key factors affecting the decision to contribute to the open-source development process: non-pecuniary benefits, future expected monetary returns, and open-source licence type. This paper argues that existing large-scale software developer surveys are inadequate for measuring the relative importance of these three factors. Previous econometric studies that collect their own unique datasets also fall short because they generally measure the importance of only one supply factor in isolation. To fill the gap, I specify an estimable dynamic programming model of joint labour supply and open-source participation decisions that can provide empirical estimates of relative importance within a single unified framework of optimal decision-making.

Key words: software, open source, labour supply, dynamic programming

JEL classification: C61, C80, J24, J44

I. Introduction

Over the past decade, there has been a phenomenal increase in the adoption of open-source software by both firms and governments. In 1996, the market share of the open-source operating system Linux, in the global server market, was roughly 6 per cent. By the year 2003, Linux's market share had reached 28 per cent. Between 1996 and 2003, Linux overtook its proprietary software competitors Unix and Netware and substantially closed the gap with the traditional leader in the sector, Microsoft Windows (see Wheeler, 2004).

*University of Southampton, e-mail: r.m.sauer@soton.ac.uk

I am grateful to Julian Morris, Eric Raymond, Corinne Sauer, and Margaret Stevens for providing insightful comments on previous drafts.
doi: 10.1093/icb/grm034

The reasons why firms and governments have increased their adoption of open-source software are generally well understood. In certain computing environments, the total cost of ownership (TCO) of open-source software can be lower than the TCO of proprietary software. Open-source software is also now considered to be of equivalent or higher quality than many proprietary software alternatives. The increasing demand for open-source software is mainly a function of falling TCO and improved program functionality (see MacCormack, 2003).[1]

In stark contrast to the demand side of the software market, the main determinants of the supply of open-source software are still unknown. The central puzzle is that most open-source software developers are volunteers who supply their labour for free, and many developers agree to have their contributions licensed in such a way that it is difficult for them to profit directly from the resulting software product. Previous research addressing this puzzle has pointed to non-pecuniary benefits, future monetary rewards, and open-source licence type as the three key factors influencing the individual's decision to contribute voluntarily to open-source development (see, for example, Lerner and Tirole, 2002). However, there is currently very little empirical evidence on the relative importance of these three supply factors. Obtaining empirical measures of relative influence can be practically important for properly predicting future open-source supply levels and assessing the impact of proposed changes in public policy towards open-source development.

In this paper, the basic theory of open-source supply contributions is outlined and the relevant econometric evidence is reviewed. It is pointed out that the empirical literature currently falls short because it generally looks at the role of monetary rewards and open-source licence type in isolation. There are also no studies, to the best of my knowledge, that attempt to identify empirically the role of non-pecuniary benefits in the supply decision. In order to fill this gap in the literature, I propose a model of joint labour supply and open-source software contribution decisions that can be used to measure the relative influence of non-pecuniary benefits, future monetary rewards, and open-source licence type in a single model. The dynamic model could be easily empirically implemented were panel data to be collected on the decisions of open-source and proprietary software developers.

The rest of the paper is organized as follows. In the next section, the basic theory of voluntary contributions to the open-source development process is outlined. In section III, the drawbacks of existing large-scale surveys are highlighted and previous econometric findings from studies that collect their own unique data sets are discussed. In section IV, the forward-looking model of software-developer employment and open-source contribution decisions, incorporating all three supply factors in a single model, is specified. The final section summarizes and concludes.

II. The basic theory of voluntary software contributions

Consider the case of a software developer who has written a fix for a bug in an existing software program.[2] Assume that the developer was originally motivated to write the patch for his own personal use of the program. In addition, assume that there is a very high degree of uncertainty regarding the value of the patch to other software developers/consumers so that there is no

[1] Governments may also prefer open-source software solutions for 'non-pecuniary' reasons. For example, open-source software can in some cases be more easily adapted to meet linguistic and cultural preferences.

[2] The following example is adapted from Raymond (1999*c*).

non-zero price at which others are willing to buy it. Under these conditions, the developer of the patch will be indifferent between keeping the patch for himself, which yields zero profits, and distributing the patch to the public for free, which also yields zero profits (assuming zero costs of distribution). The costs that the developer initially incurs to produce the patch are not relevant in the decision to release or not release because development costs are sunk.

In the above scenario, the software developer's indifference between keeping the patch private, and releasing the patch to the public for free, can be broken by assuming that there are differential expected future returns between the two options. Raymond (1999*c*) maintains that higher future returns are captured when the patch is released for free because it gives rise to reciprocal giving. That is, distributing the patch to the public for free will encourage other software developers to do the same with their own privately produced patches, and many of these patches will turn out to be useful to others in the community. Hence, releasing the patch for free is the optimal choice for each developer, and is the equilibrium outcome.

Although Raymond relies rather heavily on a social psychological notion of reciprocal giving, this assumption is not at all necessary for breaking the tie between releasing and not releasing. The same equilibrium outcome of voluntary contributions could arise if one's reputation as a skilled programmer is enhanced by distributing the patch for free. That is, by revealing one's programming code, the developer can signal his skill level, or stock of programming human capital, to the community of software developers. This could lead to differentially higher expected future returns through higher future skill prices.

This labour-market signalling function of voluntary contributions is the central notion in the work of Lerner and Tirole (2002). They also note that the signalling incentive to voluntary contributions suggests that strategic complementarities may be important. In order to signal one's skills most effectively, an open-source developer will mostly likely want to participate in open-source projects that also attract many other developers. The marginal benefit of voluntarily contributing increases with the number of developers involved in the project. Along these lines, Johnson (2002) models a developer's decision to invest effort in developing code that will become a public good, and formally illustrates the effect of a changing contributor population size.

Although the central focus in Lerner and Tirole (2002) is on the signalling role of open-source participation, and in Raymond (1999*c*) it is on reciprocal giving, the role of non-pecuniary benefits is also clearly recognized in previous research. For example, in Raymond (1999*b*) the open-source community is conceived of as a gift culture in which a developer's status in that community depends on the quality of the software gift that he/she gives to others. Another important source of non-pecuniary benefits identified by Raymond (1999*b*) is ego-gratification, or peer recognition. Developers are likened to craftsman who want others to admire their artistic style of coding. Non-pecuniary benefits can also come in the form of ideological satisfaction for those who believe software should be supplied free of charge or that Microsoft abuses its market position.[3]

While it seems plausible that higher non-pecuniary benefits and expected future monetary returns can be captured through choosing to contribute voluntarily, it is somewhat implausible to maintain that there are zero (or negligible) distribution costs to open-source participation. In fact, distributing a patch to the public for free could be very costly when there is a heavy 'regulatory burden' imposed upon submitters. For example, in some projects developers may

[3] There are also developers that receive wages from commercial firms for working on open-source projects. The reasons why commercial firms might want to pay developers to work on open-source projects are mentioned below.

be forced to comply with standards that require one to 'clean up the patch, write a ChangeLog entry, and sign the FSF assignment papers' (Raymond, 1999*c*, p. 7). The advantages derived from ego-gratification and higher future monetary rewards could be outweighed by the current and future costs of submission and distribution.

It is interesting to note in this context that Raymond (1999*c*) characterizes the Linux project as one with a relatively liberal organizational structure, and hence a relatively low cost of submission. He sees this as an important reason why Linux continues to grow and succeed. In contrast, projects with more centralized structures, such as those associated with the Free Software Foundation (FSF), have a higher cost of submission and are generally not growing as fast. Thus, variation in organizational structure and associated costs of distribution may be critical parameters in the open-source developer's optimization problem, and in the eventual market share of an open-source product.

The voluntary contributions optimization model outlined above not only provides a useful and simple theoretical framework for identifying the main factors that affect supply at the individual level, but it can also be adapted to help explain the decision of commercial firms voluntarily to open up internally developed code. Following another example in Raymond (1999*c*), suppose the internally developed software is an intermediate good in the firm's production process, e.g. an accounting package.[4] As in the individual developer's decision problem, the firm may choose to keep the initially developed code closed, or may release the code into the public domain. The initial development costs are sunk.

The main expected future benefit to releasing the code into the public domain is the receipt of programming input from hundreds of additional developers who can improve program functionality. This has been expressed in Raymond (1999*a*) as 'given enough eye balls, all bugs are shallow'.[5] However, while the firm may hope to benefit from the dispersed knowledge of developers in the wider open-source community, there is also a chance that no help will be forthcoming at all. This is because the desire of developers to participate in an open-source project initiated by a commercial firm may be weak. With firm-initiated projects, developers are less likely to reap non-pecuniary benefits related to ideological satisfaction and/or enhanced status in the open-source community. For example, contributing to improved functionality of an accounting package for a commercial firm is generally considered to be less 'challenging' than contributing to a mathematical program to be used by researchers. Developers may also intensely fear that the firm will 'hijack' the resulting software product and eventually close it off from further open-source development.

It is for these latter reasons that the form of intellectual property protection, or the licence under which a project is released, can be a critical factor in the developer's decision to supply labour to open-source development. In some cases, it may be that the only way a firm (or other project initiator) can induce developers to participate in an open-source project is to put the project under a restrictive licence such as the GNU General Public Licence (GPL). GPL is a restrictive licence because it requires that the initial code and all modifications remain freely available, that any derivative work is also licensed as GPL, and that the resulting code not be mixed with closed-source software in any re-distributed works. GPL makes commercialization of the resulting code difficult. Placing the project under GPL, rather than

[4] Raymond argues that approximately 95 per cent of all software development activities are for intermediate goods in the production process, such as accounting packages.

[5] Raymond is essentially applying ideas developed in Hayek (1945) to the case of open-source software production. That is, open-source developers have different 'local' knowledge that can be effectively tapped to the firm's benefit through the coordinating institution of open-source collaboration.

Berkeley Software Distribution (BSD) or some other less restrictive licence, can help satisfy ideological preferences, reduce the fear of hijacking, and induce greater participation from the developer community.[6]

From the firm's perspective, opening up the code, even under a restrictive licence such as GPL, can be advantageous for reasons other than improved program functionality. For example, releasing the code can help spread the risk of development. If the code remains closed, or internal to the firm, it could be costly to find suitable replacement programmers after the original in-house developers have left. Releasing the code to the open-source community can provide more continuity and fluid program maintenance, acting as a form of insurance against the deleterious effects of turnover in the market for developers (Raymond, 1999*c*). Firms can also benefit from providing complementary support and consulting services for open-source products, from increased operating system standardization which lowers the costs of providing complementary proprietary software and hardware products, and from embedding open-source components in proprietary software and hardware bundles in order to lower licensing fees. As a result of these potential benefits, several large firms have been known to fund open-source projects directly and offer salaries to open-source developers (see Berlecon Research, 2002).

On the cost side of opening up internally developed code, the firm may suffer lost profits owing to competitors in the industry being able to free-ride and benefit from the program, without having incurred initial development costs. The extent of lost profits will likely depend on the generality of the program and the industry's market structure. For example, if there is a high degree of competition in the industry then the costs of releasing the code may be large (see von Hippel, 2002; Harhoff *et al*., 2003; Henkel, 2005; Maurer and Scotchmer, 2005). On the other hand, as Lerner and Tirole (2005*b*) point out, if network effects and switching costs are considerable, then there will be little competition, the second-best software package may have only a tiny market share, and the loss in profits owing to releasing the code will be negligible.[7]

III. Empirical findings

Recently, a number of different surveys of open-source developers have become available to researchers. Most notable are the FLOSS (Free/Libre and Open Source Software) surveys which tend to over-sample developers from particular geographical regions. In the first subsection, we discuss the limited usefulness of the FLOSS-EU survey for studying the determinants of open-source software supply. In the second subsection, we review several econometric studies of the factors that influence a developer's decision to participate in the open-source development process.

(i) FLOSS-EU

FLOSS-EU was one of the first developer surveys ever to be conducted. It was administered online between February and April 2002. The questionnaire was initially posted on several

[6] The BSD licence is more conducive to commercialization of the resulting code.

[7] Firms operating in a low-transactions-costs environment may be able to mitigate free-rider costs of opening up the code by forming a consortium (see, for example, West and Gallagher, 2004).

Table 1: Descriptive statistics, FLOSS–EU (N = 2,718)

	Mean/column %
Age	27.1
Male	98.9
Single	41.4
University degree (BA, MA, or Ph.D.)	0.70
Profession	
Software engineer	33.3
Programmer	11.2
Consultant	10.4
University	9.3
Other	14.9
Student	20.1
Monthly income (€/US$)	
0	7.3
<1,000	22.1
1,001–3,000	40.1
3,001–5,000	18.6
5,001–7,500	6.0
>7,500	5.0
Weekly hours developing OS/FS	
<2	22.5
2–5	26.1
6–10	20.0
11–20	14.3
21–40	9.1
>40	7.1
Country of birth	
EU countries	0.70
North America	0.14
Other	0.16
Immigrant	0.10

Source: Gosh *et al*. (2002)

open-source/free software (OS/FS) websites and was further distributed by developers themselves. The number of respondents in FLOSS-EU is 2,784.

Tables 1 and 2 present a selected set of descriptive statistics from FLOSS-EU calculated by Gosh *et al*. (2002). Table 1 indicates that the respondents are generally young, male, single, and highly educated. The most common profession is software engineer, followed by programmer, consultant, and university employee. A little more than 20 per cent of the sample consists of university students.[8]

Approximately 70 per cent of the respondents earn less than 3,000 €/US$ a month, and 70 per cent work 10 hours or less a week developing OS/FS. Since the survey was distributed via the Internet, it reached developers in a number of different countries. For example, 70 per cent were born in an EU country. Only 10 per cent of the respondents are working, at the time of the survey, in a country other than the country in which they were born.

[8] Nearly identical percentages of students are found in the developer surveys analysed by Hertel *et al*. (2003) and Lakhani and Wolf (2005).

Table 2: Developer motivations, FLOSS–EU (n = 2,718)

	Percentage
Reason joined OS/FS community?	
Learn and develop new skills	78.9
Share knowledge and skills	49.8
Participate in a new form of cooperation	34.5
Improve OS/FS products of other developers	33.7
Participate in OS/FS scene	30.6
Software should be free	30.1
Solve problem could not solve with proprietary software	29.7
Improve job opportunities	23.9
Get help in realizing idea for a software product	23.8
Limit power of large software companies	19.0
Get a reputation in OS/FS community	9.1
Distribute not marketable software product	8.9
Make money	4.4
Do not know	1.9
Receive monetary and non-monetary rewards from OS/FS?	
No, do not earn money from OS/FS	46.3
Yes, directly; paid for administering OS/FS	18.4
Yes, indirectly; got job because of OS/FS experience	17.5
Yes, directly; paid for developing OS/FS	15.7
Yes, indirectly; but also develop OS/FS at work	12.8
Yes, directly; paid for supporting OS/FS	11.9
Yes, indirectly; other reasons	7.8
Yes, indirectly; job description does not include OS/FS work	5.2
Yes, directly; other reasons	4.4

Source: Gosh *et al*. (2002).

More directly relevant to the determinants of open-source supply, FLOSS-EU contains a number of questions related to the motivations of OS/FS developers. The top panel of Table 2 displays the percentage of respondents choosing a pre-selected set of reasons for becoming an OS/FS developer. Each respondent was allowed to choose more than one reason in the list, so the percentages can add to more than 100 per cent.

The table shows that 79 per cent of the respondents joined the OS/FS community because they were interested in learning and developing new skills. The next most frequent reason, accounting for 49 per cent of respondents, was a desire to share already existing knowledge and skills. The desire to improve job opportunities, to get a reputation in the OS/FS community, and to make money are relatively less important, but are not negligible.

The bottom panel of Table 2 displays the distribution of responses to a question on whether the developers receive monetary and non-monetary rewards for development of OS/FS. Respondents were allowed to choose more than one answer. The response frequencies reveal that more than half of the developers earn money from their OS/FS activities. Among those that earn money, a non-negligible number are directly paid for administering or developing OS/FS. A relatively high proportion claims to have secured their job as a result of their OS/FS experience. Table 2 illustrates that monetary benefits to OS/FS participation are common.

The form of the questions and responses in Table 2 make it difficult to use these data fruitfully for generalizing about key motivations. Many developers have a desire to share

existing skills, wish to learn new skills (e.g. students), and earn money either directly or indirectly from their open-source activities. However, it is not obvious how one could map these responses into metrics that would enable a researcher to disentangle the relative importance of non-pecuniary benefits, future monetary rewards, and open-source licence type. A similar interpretational problem arises with the responses to other developer surveys that have recently been conducted (see, for example, Boston Consulting Group, 2003; Haruvy *et al*., 2003; Lakhani and von Hippel, 2003; Lakhani and Wolf, 2005).

It is important to note that current developer surveys have additional drawbacks because they only reach software developers that participate in open-source projects. A control group of software developers that does not participate in open-source projects is completely absent. Another major problem is that current surveys are only cross-sectional, limiting the ability of researchers to control for unobservable characteristics that are fixed over time, such as developer ability. A survey that reached a more diverse set of software developers, that was longitudinal in structure, and that was designed with the basic economic theory of voluntary contributions in mind, would greatly help in empirically identifying the relative importance of the three key factors in the open-source supply decision.

(ii) Econometric studies

One of the most notable econometric studies to date that is related to the supply of open-source software is Hann *et al*. (2004). The authors in this paper construct an original longitudinal dataset of 147 contributors to three different Apache projects. The study aims to measure the increase in developer wages owing to the extent of contributions made to Apache projects, as well as the increase in wages due to achieving a higher rank within the Apache Software Foundation (ASF). If higher wages (skill prices) in the developer's regular employment are correlated with a higher volume of open-source contributions, then this is interpreted as a human-capital or learning effect. A higher volume of contributions proxies more open-source programming experience and greater knowledge. Any increase in skill prices deriving from a higher rank in ASF is interpreted by the authors as a signalling or sorting effect. ASF rank may be an effective means of conveying information about innate productivity levels that would otherwise be only imperfectly assessed in the market.[9]

With standard panel-data wage regressions that control for unobserved individual fixed effects—such as developer ability—the authors find that there is little return to the volume of contributions, but achieving a higher rank in ASF significantly increases wages by 13–27 per cent, depending on the rank. The conclusion is that the signalling/sorting effect is much stronger than the human-capital/learning effect in open-source Apache projects.

Although Hann *et al*. (2004), exploit an original, longitudinal data set with detailed information on open-source contributions, there are several limitations that cast doubt on the reliability of their results. As mentioned earlier in the context of existing developer surveys, the sample only includes open-source developers. Individuals who choose not to participate in open-source projects convey information about the experience and signalling value of open-source participation, yet they are not accounted for in estimation. For example, the return to ASF rank may be upwardly biased if those who do not participate in Apache projects know

[9] There are five ranks (levels of recognition) in ASF. They are: developer, committer, project management committee member, ASF member, and ASF board member. Promotion to a higher rank is awarded upon positive peer review.

that they would not experience wage increases with higher ASF ranks. Non-participants may not need to signal their productivity to the market for developers. Note that non-participants may be open-source developers that choose to contribute to open-source projects other than Apache, as well as developers that do not contribute to any open-source projects.

An additional problem that may bias the results is that ASF rank may reflect additional dimensions of open-source experience beyond the number of lines of contributed code. Therefore, the coefficient on rank may be partially absorbing the returns to open-source experience. It should not be strictly interpreted as a return to signalling/sorting.[10] Note that if open-source experience, measured as lines of code contributed, does not accurately measure true experience, then there may also be a bias in the coefficient on experience owing to measurement error. That is, attenuation bias is another possible reason why low estimated returns to open-source experience were obtained.

On the role of open-source licence type in open-source development activity, Lerner and Tirole (2005*a*) examine the determinants of licence choice using the SourceForge database which contains information on approximately 40,000 open-source projects. Lerner and Tirole (2005*a*) use the SourceForge database to run probit regressions in which the dummy dependent variable denotes either all licences in the project as highly restrictive (GPL), or in separate specifications, some licences in the project as highly restrictive. The independent variables capture project characteristics and are grouped under the headings 'development stage' (e.g. pre-alpha, alpha), 'environment' (e.g. X11, Windows), 'intended audience' (e.g. end-users, developers), 'natural language' (e.g. French, Spanish), 'operating system' (e.g. POSIX, Microsoft), and 'topic' (e.g. communications, security).

The probit results indicate that restrictive licences are more prevalent among projects that are targeted to end-users (e.g. desktop tools and games) as opposed to other developers or system administrators. This is consistent with the hypothesis mentioned earlier, in the basic theory of voluntary contributions, that restrictive licences can substitute for otherwise low non-pecuniary benefits. Applications of this type may not have a strong ego-gratification component, so that placing the project under a restrictive licence may be the only way substantially to increase utility and induce participation. Also consistent with the basic theory of voluntary contributions is the finding that restrictive licences are significantly less prevalent among projects geared towards other software developers, or projects designed for operating in commercial environments. These latter projects likely have a higher ego-gratification component and/or signalling value, and less of a need for a restrictive licence.

Although the Lerner and Tirole (2005*a*) study yields interesting insights into the determinants of a project licence, it is limited in that it remains at the level of establishing statistical correlations. Data limitations prevent accounting for unobserved project characteristics that could confound the relationship between observed project characteristics and licence choice. This makes it doubtful that they have identified any causal effects. The study also does not address the more interesting question of how licence choice affects open-source participation at the individual contributor level.

This latter question of how licence type affects open-source supply decisions is directly addressed in a paper by Fershtman and Gandal (2007). Also using the SourceForge database, the authors construct a panel of 71 open-source projects observed nine times over an 18-month period (once every 2 months). They run linear regressions of output per contributor (measured

[10] Indeed, median lines of code within ASF rank, which is used as an instrument for rank in 2SLS versions of the regressions, is strongly correlated with ASF rank in first-stage regressions. Putting the likely endogeneity of the instrument aside, the first-stage results suggest that rank is another proxy for open-source experience.

as lines of code submitted) on licence type, controlling for other project characteristics and unobserved random project effects. The results show that protecting code under more restrictive licences induces more output per contributor. The authors conclude that the significant effect of licence type is consistent with an ideological and/or status/signalling motivation to open-source participation, as hypothesized in the basic theory of voluntary contributions (see also Bonaccorsi and Rossi (2002)).

IV. The model

The review of the empirical literature in the previous section highlights how prior econometric work has focused on the influence of monetary rewards and type of open-source licence in isolation. In addition, there are no econometric studies that attempt to identify the role of non-pecuniary benefits. In this section, we propose an econometric framework that can be used to identify the relative importance of non-pecuniary benefits, monetary rewards, and open-source licence type in a single model of forward-looking optimal decision-making.

Consider a software developer that chooses among three employment states and three open-source project participation states at the start of each period t. Assume the decision problem begins at $t = \tau$ (age 18) and the terminal period, $t = T$, is the year before retirement (age 64). The length of each time period is a year. The three states in the employment choice set, denoted as K, are unemployment ($k = 0$), post-secondary schooling ($k = 1$), and full-time employment as a software developer ($k = 2$). The employment choice variable, d^e_{ikt}, is defined such that $d^e_{ikt} = 1$ if developer i chooses employment state k at time t and $d^e_{ikt} = 0$ otherwise.

The three open-source project participation states, denoted as L, are specified as no open-source project participation ($l = 0$), participation in a project licensed under BSD or other licences less restrictive than GPL ($l = 1$), and participation in a project in which all licences are GPL ($l = 2$).[11] The project participation choice variable, d^{os}_{ilt}, is defined such that $d^{os}_{ilt} = 1$ if developer i chooses open-source participation state l at time t and $d^{os}_{ilt} = 0$ otherwise. Because the developer chooses both an employment state k and a project participation state l in each time period t, the dimension of the choice set in each period is $K*L$. However, the choice set is constrained by the receipt of job offers and open-source project offers to be specified below.

The objective of the software developer is to choose an employment and project participation state in each time t to maximize the expected present discounted value of remaining lifetime utility. Remaining lifetime utility at time t for developer i is

$$V_{it}(S_{it}) = \max_{d^e_{ikt},\, d^{os}_{ilt}} E\left[\sum_{t=\tau}^{T} \delta^{\tau-t} U_{it}(\cdot)|S_{it}\right] \tag{1}$$

where V_{it} is the value function, $U_{it}(\cdot)$ is the utility flow, δ is the subjective discount factor, and S_{it} is the state space. S_{it} consists of all the factors known to the individual at time t affecting current returns or the probability distribution of future returns.

[11] A non-negligible number of open-source projects have multiple licences attached to them (see Lerner and Tirole, 2005*a*).

The maximization problem in (1) can be recast in terms of alternative specific value functions, $V_{it}^{kl}(S_{it})$, each of which follows Bellman's equation, i.e.

$$\begin{aligned} V_{it}(S_{it}) &= \max_{\{k \in K,\, l \in L\}} [V_{it}^{kl}(S_{it})] \text{ where} \\ V_{it}^{kl}(S_{it}) &= U_{it}(\cdot) + \delta E(V_{i,t+1}(S_{i,t+1}) | d_{ikt}^{e}, d_{ilt}^{os}, S_{it}), \; t < T \\ &= U_{iT}(\cdot), \qquad\qquad t = T. \end{aligned} \tag{2}$$

In the terminal period T, there is no future component to the value function and the individual maximizes current utility flow U_{iT}.

For simplicity, U_{iT} in (2) is assumed to be a linear function of consumption and open-source participation status, i.e.

$$U_{it}(\cdot) = C_{it} + (\gamma_1 + \upsilon_{i1t}) d_{i1t}^{os} + (\gamma_2 + \upsilon_{i2t}) d_{i2t}^{os} \tag{3}$$

where C_{it} is current period consumption, γ_1 is the non-pecuniary return to participating in an open-source project not licensed solely under GPL, and γ_2 is the non-pecuniary benefit of participating in an open-source project that is licensed only under GPL. γ_1 and γ_2 capture utility increases deriving from ego-gratification and ideological satisfaction that interact with licence type. υ_{i1t} and υ_{i2t} in (3) are stochastic error terms that capture heterogeneity in the appeal of open-source projects within each licensing category.

Participation in open-source projects is assumed to be constrained by the arrival of open-source project 'offers'. In each period t, it is assumed that there is a non-zero probability, denoted by λ_{1t}^{os}, that a developer receives an offer to participate in a project not licensed solely under GPL. With probability λ_{2t}^{os}, a developer receives an offer to participate in a project that is licensed solely under GPL, and with probability $1 - \lambda_{1t}^{os} - \lambda_{2t}^{os}$, the developer receives no offer and cannot choose to participate in an open-source project for that period. The open-source project arrival rates reflect prior licensing decisions by project initiators and are taken as given by the individual developer. The arrival rates could be further specified as functions of individual characteristics and current employment state in order to capture differential search intensity for projects on the part of developers. It is assumed that only one open-source project offer, at most, arrives in each period t.

Consumption in each period t is determined by a budget constraint that is assumed to be satisfied in each period and which takes the following form,

$$C_{it} = b_0 d_{i0t}^{e} + [b_1 + \varepsilon_{i1t}] d_{i1t}^{e} + w_{it} d_{i2t}^{e} - c(d_{i1t}^{os} + d_{i2t}^{os}) \tag{4}$$

where: b_0 is the current period return to being in the unemployment state and is meant to capture unemployment insurance, welfare benefits, liquidation of previous assets, and the net consumption value of leisure; b_1 is the deterministic component of the current period return to being a post-secondary school student; b_1 reflects in-school labour-market earnings and the net consumption value of schooling less tuition costs and other related expenses; ε_{i1t} is the stochastic component of schooling's current period return which captures variability in in-school labour-market earnings and other shocks to preferences for schooling; and w_{it} is the wage the individual receives as a full-time software developer and is also allowed to be

stochastic.[12] The parameter c is the consumption cost of open-source project participation which is restricted to be the same regardless of open-source licence type. The cost of project participation includes lost leisure time as well as the costs of submission/distribution of open-source code.

The wage w_{it} in (4) is further specified to be a function of education, experience, age, and an unobserved individual effect. That is, $w_{it} = w_{it}\,(x_{i1t}, x_{i2t}, os_{it}, t, \alpha_i, \varepsilon_{i2t})$ where x_{i1t} is accumulated years of post-secondary education, x_{i2t} is accumulated general experience as a software developer, and os_{it} is accumulated specific experience as an open-source software developer. The influence of age is captured by t, α_i is an unobserved individual fixed effect, and ε_{i2t} is a productivity shock assumed to be i.i.d. and serially uncorrelated. The wage function can also be augmented with observed individual characteristics, such as race and gender. However, it is generally difficult to estimate dynamic programming models of labour-force dynamics with ε_{i2t} allowed to be serially correlated.

The education and experience terms in w_{it} evolve according to the laws of motion,

$$\begin{aligned} x_{i1,t+1} &= x_{i1t} + d^{e}_{i1t} \\ x_{i2,t+1} &= x_{i2t} + d^{e}_{i2t} \\ os_{i,t+1} &= os_{it} + d^{os}_{i1t} + d^{os}_{i2t} \end{aligned} \tag{5}$$

with initial conditions $x_{i1\tau} = x_{i2\tau} = os_{i\tau} = 0$. Accumulated open-source experience is augmented by 1 year regardless of open-source project licence type. However, if empirically important, it would be easy to allow for two different accumulated open-source experience variables.

In order to minimize the number of distributional assumptions in the model, the unobserved individual effect, α_i, is specified to be stochastic with a nonparametric mass point distribution. That is, α_i is assumed to be a linear function of two unobserved 'type' dummies,

$$\alpha_i = \theta_1 A_{1i} + \theta_2 A_{2i} \tag{6}$$

where A_{1i} is a dummy variable for unobserved type 1 and A_{2i} is a dummy variable for unobserved type 2. A_{0i} is a dummy for unobserved 'type' 0. As the base type, A_{0i} is excluded from (5). In this set-up, three type probabilities, which define the discrete non-parametric distribution of α_i, are estimated along with the two non-zero location points of α_i, denoted by θ_1 and θ_2.

Assuming a standard Mincerian wage function, w_{it} can be written as,

$$w_{it} = exp(\beta_{0k} + \beta_1 x_{i1t} + \beta_2 x_{i2t} - \beta_3 x^2_{i2t} + \beta_4 os_{it} + \beta_5 t + \theta_1 A_{1i} + \theta_2 A_{2i} + \varepsilon_{i2t}). \tag{7}$$

An additional constraint imposed on the maximization problem is that a developer must receive a job offer prior to receiving a wage offer determined by (7). That is, with probability λ^e_t the developer receives a wage offer of w_{it}, and with probability $1 - \lambda^e_t$ there is no wage offer forthcoming in period t. In order to incorporate the possibility that open-source experience could have a signalling pay-off as well as a direct productivity return (as in Hann

[12] A non-zero choice probability for each of the three employment states can be generated with only two error terms. Therefore, it is not strictly necessary to specify the returns to unemployment to be stochastic.

et al., 2004), λ_t^e could be specified as a function of accumulated open-source experience at time t, as well as individual characteristics and previous employment state.

It is important to note that the model in equations (1)–(7) explicitly recognizes that education, employment, and open-source participation choices are jointly determined as well as forward-looking. Moreover, the model makes it clear which parameters are associated with current non-pecuniary benefits to open-source participation, and which parameters are associated with future monetary rewards. In addition, the influence of open-source licence type in the decision to participate in an open-source project is made explicit. The relative importance of non-pecuniary benefits, future monetary rewards, and open-source licence type can be empirically assessed in this model using the standard techniques of dynamic programming solution and structural estimation (see, for example, Keane and Wolpin, 1994, 1997; Sauer, 1998).

V. Conclusion

A review of the basic theory of optimal open-source software contributions points to three key factors affecting the decision to contribute to the open-source development process. These three factors are non-pecuniary benefits, future expected monetary returns, and open-source licence type. Unfortunately, the developer surveys that are available to researchers today are inadequate for studying the relative importance of these three key factors. Econometric studies that have collected original datasets are also limited because they generally consider the role of monetary rewards and licence type in isolation, and do not attempt to measure the influence of non-pecuniary benefits.

In order to fill the gap in the literature, this paper proposes a dynamic programming model of open-source participation decisions that could provide empirical estimates of the relative importance of non-pecuniary benefits, monetary rewards, and licence type within a single model. The model allows for three employment states (non-employment, schooling, and employment as a software developer) and three open-source participation states (no participation, participation in a project not licensed under GPL, and participation in a project licensed under GPL). In the model, developers choose an employment state and a project participation state in each period but are constrained by the arrival of employment and project participation offers. As soon as suitable panel data become available, the model could be estimated using the standard techniques employed in the literature on the solution and estimation of dynamic programming models.

In sum, economists' understanding of the motivations and supply decisions of open-source developers is still at an early stage. Higher-quality data and more comprehensive empirical models of the type proposed in this paper are needed to advance our knowledge. A better understanding of the determinants of open-source software supply is becoming increasingly important as businesses and governments rely more heavily on this 'non-traditional' software to meet their computing needs. A firmer grasp of the relative importance of non-pecuniary benefits, expected future monetary rewards, and open-source licence type would help predict future changes in open-source software supply, as well as help economists evaluate more accurately proposed changes in public policy that affect the software industry.

References

Berlecon Research (2002), *Firms' Open Source Activities: Motivations and Policy Implications*, FLOSS Final Report, International Institute of Infonomics, University of Maastricht.

Bonaccorsi, A., and Rossi, C. (2002), 'Why Open Source Software can Succeed', *Research Policy*, **32**(7), 1243–58.

Boston Consulting Group (2003), *Boston Consulting Group/OSDN Hacker Survey*, Boston, MA, Boston Consulting Group.

Fershtman, C., and Gandal, N. (2007), 'Open Source Software: Motivation and Restrictive Licensing', *International Economics and Economic Policy*, **4**(2), 209–25.

Ghosh, R., Glott, R., Kriger, B., and Robles, G. (2002), 'Free/Libre and Open Source Software: Survey and Study', University of Maastricht Institute of Infonomics and Berlecon Research, mimeo.

Hann, I., Roberts, J., Slaughter, S., and Fielding, R. (2004), 'An Empirical Analysis of the Economic Returns to Open Source Participation', Carnegie-Mellon University, unpublished working paper.

Harhoff, D., Henkel, J., and von Hippel, E. (2003), 'Profiting from Voluntary Information Spillovers: How Users Benefit by Freely Revealing Their Innovations', *Research Policy*, **32**, 1753.

Haruvy, E., Wu, F., and Chakravarty, S. (2003), 'Incentives for Developers' Contributions and Product Performance Metrics in Open Source Development: An Empirical Investigation', University of Texas at Dallas, unpublished working paper.

Hayek, F. A. (1945), 'The Use of Knowledge in Society', *American Economic Review*, **35**(4), 519–30.

Henkel, J. (2005), 'The Jukebox Mode of Innovation—A Model of Commercial Open Source Development', Technische Universität Munich, mimeo.

Hertel, G., Niedner, S., and Herrmann, S. (2003), 'Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel', *Research Policy*, **32**(7), 1159–77.

Johnson, J. P. (2002), 'Open Source Software: Private Provision of a Public Good', *Journal of Economics and Management Strategy*, **11**(4), Winter, 637–62.

Keane, M. P., and Wolpin, K. I. (1994), 'The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation and Interpolation: Monte Carlo Evidence', *Review of Economics and Statistics*, **76**, 648–72.

— — (1997), 'The Career Decisions of Young Men', *Journal of Political Economy*, **105**, 473–522.

Lakhani, K., and von Hippel, E. (2003), 'How Open Source Software Works: "Free" User-to-User Assistance', *Research Policy*, **32**, 923–43.

— Wolf, R. (2005), 'Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects', in J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (eds), *Perspectives in Free and Open Source Software*, Cambridge, MA, and London, MIT Press.

Lerner, J., and Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics*, **50**(2), 197–234.

— — (2005*a*), 'The Scope of Open Source Licensing', *Journal of Law, Economics, and Organization*, **21**(1), 20–56.

— — (2005*b*), 'The Economics of Technology Sharing: Open Source and Beyond', *Journal of Economic Perspectives*, **19**(2).

MacCormack, A. (2003), 'Evaluating Total Cost of Ownership for Software Platforms: Comparing Apples, Oranges and Cucumbers', AEI-Brookings Joint Center for Regulatory Studies, mimeo.

Maurer, S. M., and Scotchmer, S. (2005), 'Open Source Software: The New Intellectual Property Paradigm', NBER Working Paper 12148.

Raymond, E. (1999*a*), 'The Cathedral and the Bazaar', in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Cambridge, MA, O'Reilly, 19–64.

— (1999*b*), 'Homesteading the Noosphere', in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Cambridge, MA, O'Reilly, 65–112.

— (1999*c*), 'The Magic Cauldron', in *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Cambridge, MA, O'Reilly, 113–43.

Sauer, R. M. (1998), 'Job Mobility and the Market for Lawyers', *Journal of Political Economy*, **106**, 147–71.

von Hippel, E. (2002), 'Open Source Projects as Horizontal Innovation Networks—By and For Users', MIT Sloan School of Management Working Paper No. 4366-02.

West, J., and Gallagher, S. (2004), 'Key Challenges of Open Innovation: Lessons from Open Source Software', San Jose State College of Business, mimeo.
Wheeler, D. (2004), 'Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers!', available at http:///www.dwheeler.com/oss_fss_why.html (accessed 12 December 2004).