

# Firms as Incubators of Open Source Software

Rajiv Dewan\*, Marshall Freimer\*, Amit Mehra\*\*

\*William E. Simon School of Business Administration, University of Rochester, Rochester, NY 14627 USA

\*\*Indian School of Business, Hyderabad, India

dewan@simon.rochester.edu, freimer@simon.rochester.edu, amit\_mehra@isb.edu

Many successful open source projects have been developed by programmers who were employed by firms but worked on the open source projects on the side due to economic incentives like career improvement benefits. Such side work may be a good thing for the employing firms too if they get some strategic value from the open source software and/ or if the productivity of the programmers on these projects improves due to learning by doing effects. However, the programmers may work more or less on these projects than what is best for the firms. To manage the programmers' effort the firms set appropriate employment policies and incentives. These policies and career concerns then together govern the programmers' effort allocation between the open source and proprietary projects. We examine this relationship using a variant of the principal agent model. We derive and characterize the optimal employment contracts and show that firms either offer a bonus for only one of the two projects or do not offer any bonuses. But if attractive alternate employment opportunities are available, they change their strategy and may offer bonuses for both projects simultaneously.

---

## 1. Introduction

Brian Behlendorf was employed as a web administrator at *Wired* when he started to work on the open source project that became Apache, the web server that today has almost 70% market share (news.netcraft.com). Similarly, Larry Wall was employed at Unisys as a programmer when he invented the Perl language and wrote an open source interpreter for it. Today many consider Perl to be the “glue that holds the web together.” Many open source programmers are similar to Behlendorf and Wall in that they have a “day” job while they “moonlight” on open source projects. 55% of programmers in a survey by Lakhani and Wolf (2005) confessed to contributing open source code in their work time. This is apparently a cause for worry for firms because their employees have extraneous opportunities to contribute time/energy to open source projects which is not what they were hired to do.

However, firms like IBM and Sun Microsystems have their programmers contribute code to Eclipse and OpenSolaris open source projects, respectively. A study conducted by United Nations University UNU-MERIT (2006) finds that contributor firms to open source are not marginal players in the economy. Their sample of 158 open source contributing firms indicates that such firms have

in total 530 thousand employees with a total annual revenue of Euro 231.4 billion. Clearly several firms benefit from the development of certain open source projects.

To examine the incubation of open source software in firms, we first highlight some reasons due to which open source programming activity may be beneficial for the firms. A common open source software based business model is to sell complements to some popular open source software. The success of such an strategy depends upon development of the base software that is open source (Fink (2002)). Consequently, firms following such business models may endorse or encourage their employees to work on selected open source projects.

Another reason due to which firms may encourage open source programming activity is the learning by doing benefits that their programmers may get by working on open source projects. This improves the programmers' performance on the firm's projects and thus is beneficial to the firms. Writing on the value of human capital, Arrow (1962), Becker (1962) and Blaug (1976) explain that learning is very valuable and that it takes place in one of two ways: *Learning by doing* and *learning by investing in training*. This literature also recognizes that the two forms of learning are not substitutes, in that they may provide mutually exclusive benefits (Killingsworth (1982)). The learning by doing benefit of open source software has been examined through survey based papers by Lakhani and Wolf (2005), Hars and Ou (2002) and Hertel, Niedner, and Herrmann (2003). All these papers find overwhelming evidence that skill building is an important reason due to which programmers choose to contribute to open source projects. von Krogh, Spaeth, and Lakhani (2003) suggest reasons for why learning by doing happens due to open source participation. They observe that peers in the project community, software users, and interested outsiders who are involved in open source projects help evaluate code submissions, find faults in programming and often suggest changes to improve the performance of the code. This interactive process improves both the quality of code submission and overall programming skills of the participants. Industry insiders like Moody (2002), Raymond (2001) and Wayner (2000) also confirm that an active peer review process forms the foundation on which the learning by doing from open source projects is based. Lakhani and von Hippel (2003) report on the motives of this peer review process. Besides helping the community and altruism, they find that programmers help each other in order to find out about the problems others have faced. This learning helps them to manage and update their own web sites and software code. A report compiled for the European Communities, United Nations University UNU-MERIT (2006), delved deeper into the issues of skill building through open source projects. It reports that programmers indicated the programming skills learnt open source participation can

even compensate for lack of formal degrees. In fact, many respondents felt that programming skills are better learnt through open source activities than through formal courses.

Not only the programmers, but the firms also agree with the importance of learning from open source projects. The United Nations University UNU-MERIT (2006) report finds that employers believe that complex technical, management and legal skills (writing reusable code, understanding of licensing, coordination of work with other people) are better learnt through open source activities. 16 percent of companies in sectors such as retail, automobile, tourism and construction allow their programmers to get involved in open source activities even when their business models do not derive strategic value from open source software. Thus there is sufficient evidence that several firms benefit from their programmers' learning skills from participating in open source projects. In a complementary finding, Rossi and Bonaccorsi (2006) show through their survey that one of the important motivations for firms to let their programmers participate in open source projects is to study the code written by other programmers and use it to develop new programs and solutions. Tim O'Reilly, one of the experts on open source espoused a similar reason to exhort Microsoft to contribute to open source software<sup>1</sup>. Thus learning from open source projects directly increases the value of the programmer's work for the commercial firms. This is not surprising in the light of Schilling, Vidal, Polyhart, and Marangoni (2003) who use an experimental study and Boh, Slaughter, and Espinosa (2007) who use empirical data to show that diverse experience in related systems improves programmers' learning and thereby facilitates increased productivity. The value of learning from open source for the firms can also be explained through the value of external knowledge (Menon and Pfeffer (2003)). That is why firms resort to activities such as hiring outside consultants and acquisitions. We therefore infer that learning by doing open source software projects may offer human capital development beyond what is provided through classroom training activities and learning by doing from the firm's projects themselves.

Besides learning from open source projects, programmers learn from working on firm's projects as well. Reagans, Argote, and Brooks (2005) find that individuals working inside an organization become more productive as they learn from cumulative experience of others and also learn to coordinate better as they discover who knows what and trust each other more. Thus programmers learn due to their work in an organization and this is useful in improving programmer productivity. Further, Boh, Slaughter, and Espinosa (2007) show that programmer learning improves the most from experience in a specific system. Hence working in proprietary projects significantly improves skills. Finally, since programming skills can be construed as industry specific human capital (Neal

<sup>1</sup> [http://oreilly.com/news/osconint\\_0601.html](http://oreilly.com/news/osconint_0601.html)

(1995)), any programming skills that the programmers learn are useful in improving their productivity in both the open source and the firm's projects. Accordingly, our model incorporates the effect of learning by doing on programmers' performance from both the open source and the firm's projects. In this way, we contribute to the literature on human capital formation.

Multi-tasking on open and proprietary projects by programmers should not pose any problems since there is ample evidence that firms have increasingly adopted organizational structures that encourage multi-tasking (Lindbeck and Snower (2000), Pautrel (2004), Cartensen (2002), Lindbeck and Snower (1996), Breshnahan, Brynjolfsson, and Hitt (2002), Caroli and Reenen (2001)). This implies that the firms are amenable to sustaining open and closed source activity by the programmers. Learning by working on open source projects is costless for the firm beyond the opportunity cost of its programmers' time spent on these projects.

Besides learning by doing, career incentives also motivate programmers to contribute to open source projects. Proof of high talent leads to high wages in the labor market (Weiss (1995)). For a number of reasons open source projects may be better than a firm's proprietary project for conveying programmers' talent to the labor market. Anyone can examine the code base of an open source project and directly evaluate a programmer since open source projects traditionally document each contribution. In addition, the mere acceptance of a programmer's contribution in the code base is in itself a proof of high quality work because only contributions that pass the scrutiny of open source project leaders are included. This kind of evaluation is generally not possible for proprietary projects.

Extant literature on career concerns deals with issues of optimal contract design to induce workers to make better corporate investment choices as in Milbourn, Shockley, and Thakor (2001), or better effort choices in presence of moral hazard as in Holmstrom (1999), Gibbons and Murphy (1992) and Lal and Srinivasan (1993). Hann, Roberts, Slaughter, and Fielding (2006) empirically confirm that enhancement in career is one reason why programmers contribute to open source projects. Their study was based on the analysis of the Apache web server project and they found that successful open source participation, measured by a higher rank in the hierarchy of programmers in the project, translated into higher wages. The work in this paper contributes to the career concerns literature in a setting where open source software provides additional means through which programmers' talent may be conveyed to the labor market.

To summarize the above discussion, both the hiring firms and their employed programmers have economic incentives to work on open source projects. Firms value open source work due to the strategic value they may derive from these projects and the learning by doing benefits to their

programmers which helps improve their productivity. Programmers value open source work due to career incentives and the learning by doing effects which helps them to do a good job and hence get better compensation. However, the incentives of the firm and the programmers are not aligned. Consequently, the programmer may choose to devote effort/attention in the two projects that are not optimal from the firm's perspective. Since programming is a task that requires significant autonomy and creativity, the firm cannot easily monitor (Kirsch (1996)) the effort/attention devoted by programmers to each of the projects. To the extent the firm cannot perfectly monitor the effort/attention division, this *effort division* results in lower net benefits for the firm and a *moral hazard* situation for programmers. In order to mitigate this problem, the firm may provide incentives to programmers through setting different bonuses for good performance in the proprietary and open source projects. Such bonuses will then induce the programmers to devote effort/attention to the projects that are favorable from the firm's perspective. The study of this incentive problem for the firm forms the central theme of this paper. In order to analyze the different phenomenon that arise from open source participation, we construct a principal-agent model in which a programmer decides on effort allocation between a proprietary project and an open source project. In this variant of the principal-agent model, we explicitly include effort division, learning by doing, and career concerns.

We find the marginal effect of effort to be a key determinant of contract choices offered by the firm. For high expected marginal revenues with respect to effort in the closed source project, the firm offers a contract that provides a bonus for good performance in the closed project only. For low levels of expected marginal revenues, the firm offers a contract that provides a bonus for good performance in the open project only. For moderate levels of expected marginal revenues, the firm does not offer any bonus for either project. If the programmers have alternate employment options additional compensation may be required to make the firm's employment as attractive as these alternate opportunities. In such cases, the firm may offer a contract that provides bonuses for both the open and closed projects. The contract in this situation is socially optimal since it maximizes the total value created by the programmer. Other contracts do not have this feature.

We find that the degree to which the proprietary project is open, i.e. the amount of information about individual contributions in proprietary projects that is revealed to the market may be an important factor influencing the total value created for the firm. An intermediate value of the degree of openness of the proprietary project is sometimes optimal. However, when alternate employment options are good requiring the firm to give bonuses for both the open and closed projects, the impact of this factor on firm value vanishes.

While we focus on learning by doing and career incentives of programmers as drivers of open source contributions, alternative explanations that justify programmers contributions to open source also exist. Johnson (2002) analyzes open source contributions in the spirit of the public good literature in economics. Other reasons that have been advanced are the entertainment value of working on what programmers might like doing, and the ego gratification when their contributions are recognized in the community of programmers (Lerner and Tirole (2002)). Some factors may be stronger than others in driving open source participation depending on programmer demography. Our work is directly relevant in case learning by doing and career incentives are the main forces (these are the likely drivers for fresh programmers). Even when the other factors are important, their main effect is that open source activity may be more attractive for the programmers than what we model. The firm can therefore adjust its contracts by perhaps paying lower bonuses.

In summary, we consider the issue of incentives of programmers to work on open source programming projects while employed, and analyze the consequent optimal contract choice by a hiring firm which may have some strategic value from open source projects. The next section describes an analytical model for examining these questions. Section 3 provides an analysis of the model and specifies the choice of bonus contracts for the firm. Section 4 examines the impact of degree of openness of the closed project on firm value. Finally, Section 5 provides concluding remarks. All lemmas and details of proofs of the propositions are in the Appendix.

## 2. Model

### 2.1. The Market for Software Development

The labor market for programmers consists of several firms. Each firm hires programmers to do programming projects at the beginning of each of an infinite succession of time periods. A programming project is an individual task that is assigned to a programmer and the programmer is solely responsible for creating the output to complete that task. Now we provide some details about both sides of this market: Firms and programmers.

**2.1.1. Programmers** A programmer may be either “talented” or “untalented”. To simplify the information structure of the problem, we assume that success, or good outcome in a project is only possible if the programmer is talented. This is not an unreasonable assumption for the software industry; software industry experts have pointed out that the productivity of talented programmers is much higher than that of average programmers.<sup>2</sup> Further evidence comes from a

<sup>2</sup> Jim Clark, one of the founders of Silicon Graphics hired Pavan Nigam because, “...the difference between a great software guy and an OK software guy is huge. A great software guy is worth 10 times an OK software guy.”- New York Times Magazine (Oct 10, 1999)

study in Goleman (1998), where it was found that the top 1% of programmers were 1272% more efficient than the average. Thus it can be reasonably expected that in a given time period a talented programmer can achieve a better output compared to an average programmer.

The information on the talent of a programmer may/ may not be known and depends on how long the programmer has been in the programming industry. A programmer who has freshly entered the labor market after completing formal education is termed as a “rookie”. While rookie programmers know their own capabilities, they are relatively unaware of industry requirements. Therefore, they are unable to ascertain how well their capabilities are suited for the requirements of the industry. In other words, their information set is insufficient to let them conclude whether they are talented or not with certainty. Similarly, while a hiring firm in the labor market is informed about the industry requirements, it has little information about the capabilities of an individual rookie programmer. Thus the firm is also unable to conclude how well a particular individual is likely to perform. Due to these information asymmetries, we assume that neither the rookie programmers nor the labor market know whether a particular rookie is talented or not. We assume that there is a common prior probability  $m$  with which a rookie programmer may be talented. Analysis of a labor market with a similar description has been made earlier by Holmstrom (1999) where he studied how the incentives of workers to exert effort change along their career path. We further assume that a rookie programmer when employed can create a minimum value of  $g$  per period (i.e. both talented and untalented programmers create this value). Further, the market wage for rookies is normalized to zero.

As rookie programmers are employed and their contributions evaluated, the hiring firm, the programmers themselves, and other firms in the market find out talent of the programmers. The market wage for programmers who are known to be talented in the labor market is  $w_t > 0$ . Thus, if the rookie programmers prove themselves to be talented, then their wage increases from zero to  $w_t$ . This wage premium provides a career incentive to the rookies to convey their talent to the labor market.

**2.1.2. Firms** A firm can get an expected value of  $w_t$  by employing a talented programmer for one period<sup>3</sup>. However, if such a programmer is employed in low value projects, the value generated may be lower than  $w_t$ .

<sup>3</sup>This amounts to assuming that the market for talented programmers is a ‘sellers’ market where all the surplus goes to the programmers. In a general equilibrium in this labor market, the expected value of the projects is  $w_t$  and this is also the market wage of the talented programmers. This assumption allows us to focus on the contract for rookies and has earlier been used by Holmstrom (1999).

At the beginning of the game, the focus firm has a commercial closed project and a synergistic open source project with learning by doing benefits between the two projects. A successful outcome (meaning that the program provides all the user demanded functionality, does so without using too many computing resources, and is easy to maintain) in the open and closed projects creates a value  $g_o$  and  $g_c$  respectively. These values are over and above the minimum value  $g$  that is created by employing any programmer. We are interested in cases when  $g$ ,  $g_o$  and  $g_c$  are not so large that the firm will directly hire a programmer with proven talent at the high market wage ( $w_t$ ). We instead examine the case when these values are more moderate so that the firm will take a chance on hiring a rookie for its projects.

## 2.2. Revelation of Programmer Talent to the Market

The revelation of a rookie programmer’s talent depends on two events: Success in open and closed projects and observation of this success by the market. Here we discuss the probabilities of both these events.

**2.2.1. Probabilities of success in open and closed projects** The probabilities of the different outcomes of the two projects conditional on having a talented programmer are displayed in Table 1. A highly successful outcome is classified as *good* and the other outcome as *bad*.

	Good in Open	Bad in Open	<i>Marginal Probability</i>
Good in Closed	$p_{gg}$	$p_{gb}$	$p_c = p_{gg} + p_{gb}$
Bad in Closed	$p_{bg}$	$p_{bb}$	$1 - p_c$
<i>Marginal Probability</i>	$p_o = p_{gg} + p_{bg}$	$1 - p_o$	

**Table 1** Joint and Marginal Probabilities of Outcomes for a Talented Programmer

Recall that previously we have assumed that the probability of success is zero in all projects for untalented programmers.

All of these joint and marginal probabilities are functions of effort division, with fraction  $x$  of the effort being devoted to the closed source project and  $1 - x$  of the effort to the open source project<sup>4</sup>. As discussed in the Introduction, we assume that this effort division is made by the programmer and is neither observable nor contractible by the firm in its employment contract with the programmer.

<sup>4</sup>We use a budget type model of effort allocation in which the programmer selects between effort devoted on the closed source and the open source projects. Other potential uses of time and effort, such as recreation, are not considered. Others papers in contract/ agency theory like Lindbeck and Snower (2000) also make this assumption in a somewhat similar modeling set-up with two projects.

The marginal probabilities ( $p_c$  and  $p_o$ ) are dependent on both the effort division  $x$  and learning by doing from the two projects. As the effort in a project increases, the positive impact of effort and learning by doing from this project on its marginal probability of success reduces due to diminishing returns. Further, increase in effort in this project implies reduction of effort in the other project. Consequently, there is a negative impact on the marginal probability of success of this project due to reduction in learning by doing from the other project. Due to the diminishing returns phenomenon, this negative impact is increasing as the effort in the other project reduces. If this negative impact is strong enough, the slope of the marginal probability of success will eventually become negative. Clearly, this probability will be at its maximum at intermediate effort levels in such situations.

Similar to the above, for intermediate values of  $x$ , the probability of failure in both projects ( $p_{bb}$ ) will be low because of better learning opportunities and balanced effort in both projects. At more extreme values of  $x$ , the imbalance in learning opportunities and effort division will cause an increase in this probability.

While both effort and learning by doing are inputs in determining the marginal probability values, we assume that the two projects are sufficiently differentiated and it is not the case that learning benefits from one project reduce effort requirement very significantly in the other. Such a situation may indeed happen if, for example, a piece of code developed for one project can be reused in the other project directly with minimal changes. The firm may lose valuable intellectual property if the code developed for closed source project is re-used in the open source project. On the other hand legal issues due to the open source license would arise if the code developed for the open source project is reused in the closed source project without making its source code open. We do not consider such situations since firms are unlikely to risk hiring programmers to do both projects if this is the case. Given this focus, one would expect marginal probability of success in a project would reach a maximum at an effort division that favors that project.

Putting together the above we state the following assumption.

*ASSUMPTION 1. Marginal probabilities of success,  $p_c$  and  $p_o$ , for the closed and open source projects, respectively, and the probability of obtaining any success,  $p_{gg} + p_{gb} + p_{bg} = 1 - p_{bb}$  are doubly differentiable wrt  $x$ , and strictly concave, i.e.,  $p''_c, p''_o < 0$ , and  $p''_{bb} > 0$  for  $0 \leq x \leq 1$ . Further, probabilities  $p_c$  and  $p_o$  are maximized at  $x_c$ , and  $x_o$ , respectively, such that  $0 < x_o < x_c < 1$ .*

One implication of Assumption 1 is that  $p'_c > 0 \forall x < x_c$  and  $p'_c < 0 \forall x > x_c$ . Similarly,  $p'_o > 0 \forall x < x_o$  and  $p'_o < 0 \forall x > x_o$ .

Our last assumption about probabilities requires that an increase in effort on the closed project increases  $p_{gb}$  and decreases  $p_{bg}$ . Such behavior is expected when effort is the only driver of project success. This assumption says that the moderating effect of learning by doing on probabilities is not too significant. Together, assumptions 1 and 2 implicitly state lower and upper bounds of the impact of learning by doing.

ASSUMPTION 2.  $p'_{gb}(x) > 0$  and  $p'_{bg}(x) < 0$  for  $0 < x < 1$ .

### 2.2.2. Difference in observability of outcomes of open and closed source projects:

Open and closed source projects differ in the information they make available to the labor market. Consequently there is a difference in what the labor market finds out about the programmers' talent from observing the closed and the open projects.

The outcome in the open source project is visible to all. This is because the open source community has a tradition in which the contributions of individual programmers are explicitly recognized. For example, the web site <http://sourceforge.net/projects/inkscape/> lists the programmers who have contributed to an open source drawing tool called *Inkscape*, which is being developed as an open source alternative to Adobe Illustrator and Corel Draw etc. It is possible to get information about individual contributions from facilities like CVS/SVN commits provided on the project web pages. Further, as the community itself selects code fragments to be included in the released product, the acceptance of a contribution is in itself revealing of a programmer's talent. Thus the labor market gets a clear insight into the contribution made by each programmer.

Closed source, by its very nature, is much less revealing. Not only is the code not available for evaluation, the individual contribution is rarely made public. To focus on this difference between open and closed source projects we assume that the outcome of the open source project, success or failure, is observed publicly by all. In contrast, we assume that success in a closed source project is imperfectly and asymmetrically visible. While the hiring firm and the programmer always observe the outcome of the closed source project, the market gets to see the successful outcome of this project with probability  $p$ .

At this point, we define a useful class of probability functions as follows:

DEFINITION 1.  $p_r(z) = p_o + zp_{gb} \forall z \in [0, 1]$

From Assumptions 1 and 2 it is easy to see that  $p_r(z)$  is concave<sup>5</sup> and that  $p_r(z)$  is maximized at  $x_r(z)$  such that  $x_o < x_r(z) < x_c$ <sup>6</sup>.

<sup>5</sup>  $p''_o + zp''_{gb} = z(p''_{gg} + p''_{bg} + p''_{gb}) + (1-z)p''_o < 0$ , since coefficients of  $z$  and  $1-z$  are both negative.

<sup>6</sup>  $p'_r(z)|_{x=x_o} = p'_o|_{x=x_o} + zp'_{gb}|_{x=x_o} = zp'_{gb}|_{x=x_o} > 0$ , since  $p'_o|_{x=x_o} = 0$  and  $p'_{gb}|_{x=x_o} > 0$  from Assumptions 1 and

Using  $p_r(z)$ , the probability that a talented rookie programmer is revealed to be so to the market is:

$$p_t = p_r(p)$$

Consider the following example to illustrate all the probabilities discussed above:

EXAMPLE 1. Let  $\pi(x) = ex$  (for  $0 \leq e \leq 1$  and  $0 \leq x \leq 1$ ) be the probability of success in any one project if proportion  $x$  of time is spent on it and there is no learning value of one project for the other.

Suppose that a talented programmer spends effort  $x$  on the closed project and thereby spends leftover effort  $1 - x$  on the open project. The joint probabilities of the various outcomes for a talented programmer taking into account the learning effect  $d$  are:

$$\begin{aligned} \text{Prob}[\text{Good in closed, good in open}] &= p_{gg}(x) = (1 + 2d)\pi(x)\pi(1 - x), \\ \text{Prob}[\text{Good in closed, bad in open}] &= p_{gb}(x) = (1 - d)\pi(x)(1 - \pi(1 - x)), \\ \text{Prob}[\text{Bad in closed, good in open}] &= p_{bg}(x) = (1 - d)(1 - \pi(x))\pi(1 - x), \\ \text{Prob}[\text{Bad in closed, bad in open}] &= p_{bb}(x) = 1 - p_{gg}(x) - p_{gb}(x) - p_{bg}(x) \end{aligned}$$

Note that  $d = 0$  corresponds to the no learning case. Further, the probabilities make sense only when  $d < 1$ . Using the definitions above we can easily work out the marginal probabilities of success,  $p_c$  and  $p_o$ . It is easy to verify that these probabilities agree with Assumption 2 and satisfy Assumption 1 if  $\frac{1}{1+3e} \leq d \leq 1$ . Thus, as explained earlier, the assumption put bounds on the extent of learning effects.

The marginal probabilities,  $p_c$  and  $p_o$ , and the probability of revelation of talent,  $p_t$ , are illustrated in Figure 1 for  $d = 0.6$ ,  $e = 0.5$  and  $p = 0.5$ .

### 2.3. Surplus for Rookie Programmers and the Firm

Recall that we are considering situations where the firm initially hires a rookie programmer for a single period for doing open and closed source projects. The rookie may be rehired in future either by this firm or by other market firms based on her performance. In this section, we focus on getting the expressions for the total surpluses of the rookie and the firm. In writing these surpluses, we use  $\delta$  as the discount rate per period. At this point, we introduce Table 2 to summarize various symbols and their definitions.

2 respectively. Further,  $p'_r(z)\Big|_{x=x_c} = -(1 - z)p'_{gb}\Big|_{x=x_c} + p'_{bg}\Big|_{x=x_c} < 0$ , since  $p'_{bg}\Big|_{x=x_c} < 0$  and  $p'_{gb}\Big|_{x=x_c} > 0$  from Assumption 2 and  $p'_c\Big|_{x=x_c} = 0$  from Assumption 1. These imply that  $p_r(z)$  is maximized at some  $x_r(z)$  where  $x_o < x_r(z) < x_c$ .

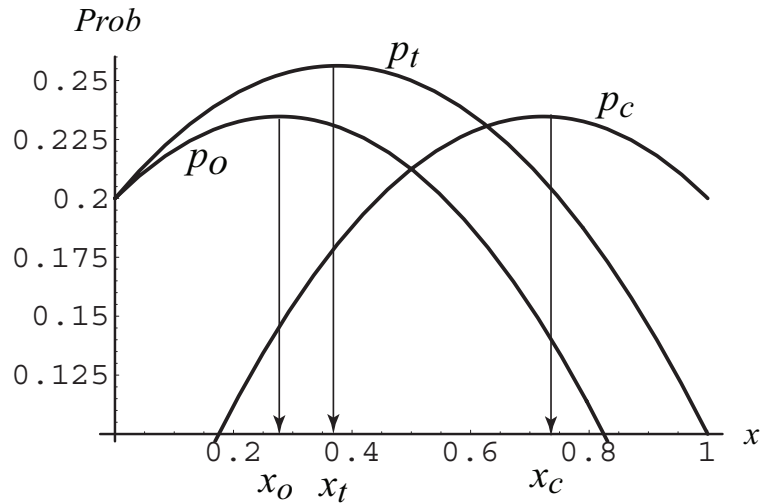


Figure 1 The marginal probabilities

Symbol	Definition
$x$	Fraction of effort in the closed source or proprietary project
$x_c$	Fraction of effort in the closed project at which $p_c$ is maximized
$x_o$	Fraction of effort in the closed project at which $p_o$ is maximized
$w_c$	Bonus for success in proprietary project
$w_o$	Bonus for success in the open source project
$m$	Probability that a rookie programmer is talented
$g$	Minimum value created by any programmer in one period
$g_c$	Value created due to success in the firm's first period proprietary project
$g_o$	Value created due to success in the first period open source project
$w_t$	Maximum expected value created by a talented programmer in one period, market wage of talented programmers
$u$	Minimum utility for the programmer to accept the firm's employment
$p_o$	Marginal probability of success in the open project
$p_c$	Marginal probability of success in closed project
$p_t$	Probability that a talented rookie programmer is revealed to be talented in the market
$p$	Probability that the market observes the successful outcome in the closed project
$EW$	Expected bonus for the rookie in period 1
$ES$	Expected surplus for the rookie programmer
$EV$	Expected value for the firm
$EV_t$	Total value created for the firm and the programmer
$MR$	Marginal Revenue for the firm as a function of $x$
$\delta$	Discount rate per period

Table 2 Parameter and variable definitions

**2.3.1. Surplus for rookie programmers** Suppose that the firm offers a contract that provides non-negative bonuses  $w_c$  and  $w_o$  for good outcomes in the closed and open source projects, respectively, to a rookie in the first period. The expected bonus for the rookie in this period is:

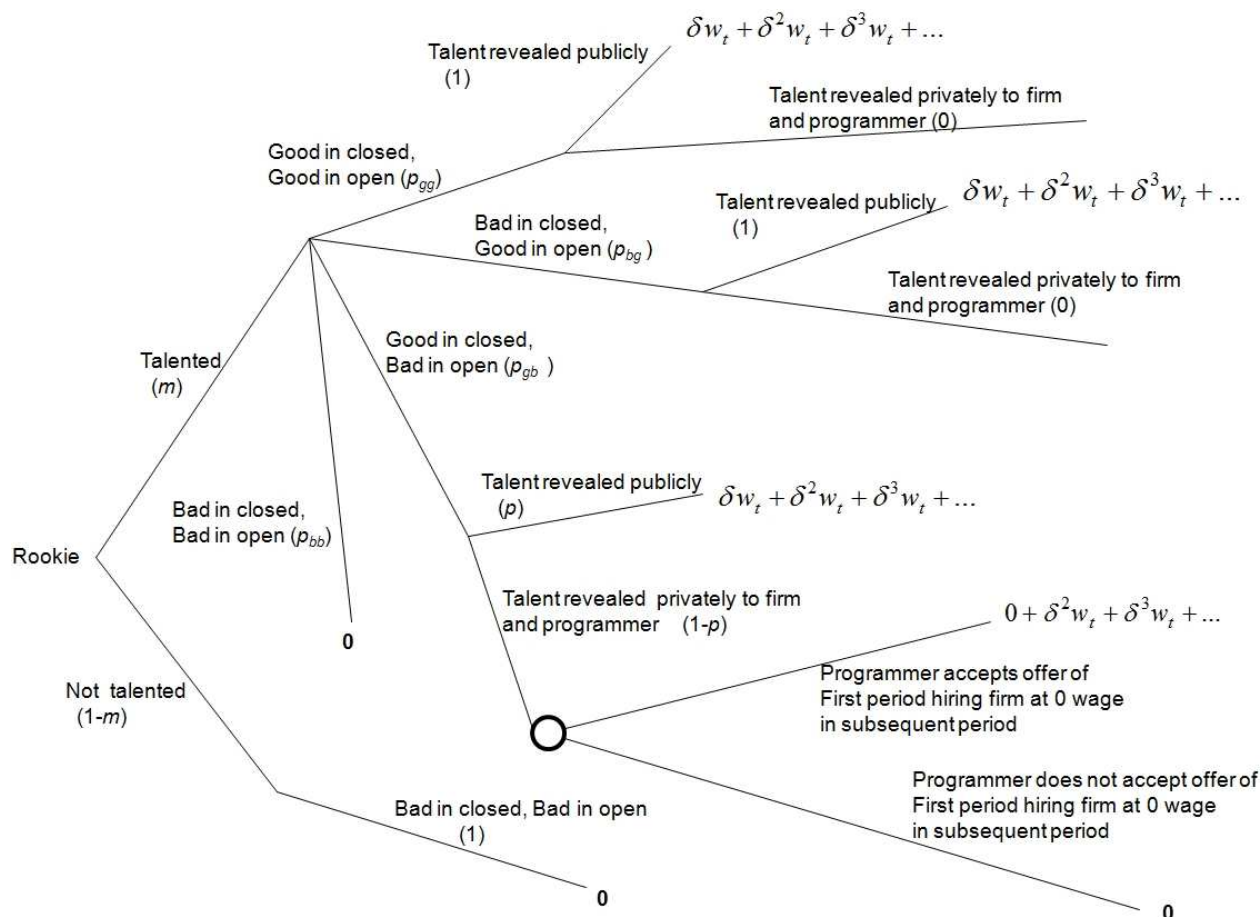


Figure 2 Future Surplus for Rookie Programmer

$$EW = mp_cw_c + mp_o w_o \quad (1)$$

The rookie programmer’s future employment opportunities are contingent on the outcomes in the first period since these outcomes may reveal whether the programmer is talented or not, publicly (information becomes available to programmer, first period hiring firm and all market firms), or privately (information becomes available to programmer and first period hiring firm while the market firms are excluded). The complete tree depicting the events and decisions is shown in Figure 2. The events are noted at each branch and the probabilities of the events are noted in parenthesis. The future surpluses earned by the programmer are noted at the culmination of a sequence of events. If a particular sequence of events is impossible (probability 0 event), then the surpluses for that sequence are omitted. The circle in the figure depicts a decision node; the programmer’s decision at this point influences future surpluses.

If a rookie is untalented, she fails in producing a good outcome in both projects. Further, even if the rookie is talented, she fails in producing a good outcome in both projects with probability  $p_{bb}$ .

Thus a rookie who produced a bad outcome in both projects can only be talented with probability  $\frac{mp_{bb}}{(1-m)+mp_{bb}}$ . One can easily show that  $\frac{mp_{bb}}{(1-m)+mp_{bb}} < m$ . Thus a fresh rookie has a greater probability of being talented than a rookie who failed in producing a good outcome in both projects. Further, the market wage of rookies is zero. Thus all firms will always prefer to employ fresh rookies over failed rookies. Due to this, the failed rookies will have no future earning potential. Accordingly, the lowest two branches in the event tree in Figure 2 show future earnings of 0.

If the rookie programmer is talented, has succeeded in one or more projects and her success is revealed to the labor market, then she can find employment in all subsequent periods with a wage of  $w_t$  in each period. Thus the present value of the surplus is  $\delta w_t + \delta^2 w_t + \delta^3 w_t + \dots$ . This surplus is earned with probability  $m(p_{gg} + p_{bg} + pp_{gb}) = mp_t$ .

An interesting situation arises when a talented programmer succeeds only in the closed project, and this information does not become public. Will the market firms be interested in hiring such a programmer? Note that the market firms cannot distinguish between these programmers and those who genuinely failed in producing a good outcome in both projects. Thus if the market firms pick a programmer from this pool, the probability that this programmer is talented is  $\frac{m(1-p)p_{gb}+mp_{bb}}{(1-m)+m(1-p)p_{gb}+mp_{bb}}$ . It is easy to show that  $\frac{m(1-p)p_{gb}+mp_{bb}}{(1-m)+m(1-p)p_{gb}+mp_{bb}} < m$ .<sup>7</sup> Thus the market firms can get talented programmers from the fresh rookie pool with a higher probability at zero wages. Therefore these firms will never give an offer to such failed programmers. In other words, the outside opportunities for these programmers are zero in this period. As a result these programmers will take an offer from the first period hiring firm even at zero wages. Note that failure to publicly show a successful output results in a penalty since such programmers are reemployed at zero wages while their peers who could publicly show a successful output are reemployed at  $w_t$ . However, this is not a permanent problem since the act of rehiring by the first period firm in the subsequent period informs the rest of the market firms that such a programmer was indeed talented. Therefore, the programmer is recognized as a talented programmer after a lag of one period and she then starts to earn the wage  $w_t$  per period implying that the present value of future earnings are  $0 + \delta^2 w_t + \delta^3 w_t + \dots$

According to the above discussion, the programmer's expected surplus from the current and all future periods is:

$$\begin{aligned} ES &= EW + mp_t(\delta w_t + \delta^2 w_t + \dots) + m(1-p)p_{gb}(0 + \delta^2 w_t + \delta^3 w_t + \dots) \\ &= EW + mp_t \frac{\delta w_t}{1-\delta} + m(1-p)p_{gb} \frac{\delta^2 w_t}{1-\delta} \end{aligned} \quad (2)$$

<sup>7</sup> The inequality can be rewritten as  $m(1-p)p_{gb} + mp_{bb} < m(1-m) + m^2((1-p)p_{gb} + p_{bb}) \implies (1-p)p_{gb} + p_{bb} < 1$ , which must be true since  $0 < 1-p < 1$ ,  $0 \leq p_{gb} \leq 1$ ,  $0 \leq p_{bb} \leq 1$  and  $0 \leq p_{gb} + p_{bb} \leq 1$ .

Examining Equation 2 we note that the market wage  $w_t$  for talented programmers sets up an incentive for the programmers to make effort choices in the current period to increase the chance of revealing her talent in future. This *career concern* is similar to that in Holmstrom (1999).

**2.3.2. Firm surplus** The expected value of the first period projects to the firm for which it hires rookies is  $g + mp_c g_c + mp_o g_o$ . This is not the only value created for the firm when it rehires programmers in subsequent periods. With probability  $m(1-p)p_{gb}$  the firm will be in a situation where it knows that its current employee is talented but the labor market does not. As pointed out in the last section, this allows the firm to re-hire such programmers at zero wages for the subsequent period. In order to induce these programmers to work and avoid shirking, the firm must give a bonus  $\epsilon > 0$  that is contingent on a successful outcome in this period. Even if  $\epsilon$  is made arbitrarily small, the programmers maximize their surplus by maximizing the chances of producing a successful outcome. Thus shirking can be avoided at negligible cost. Consequently, the firm gets an expected rent of  $m(1-p)p_{gb}w_t$  in the next period. Putting these terms together and considering the wage paid to the programmer, we get the total expected surplus, in net present value, for the firm by hiring a rookie programmer to be:

$$EV = g + mp_c g_c + mp_o g_o + m(1-p)p_{gb}\delta w_t - EW \quad (3)$$

Assuming that  $m > 0$ , we drop it from expressions of  $ES$ ,  $EW$  and  $EV$  in the following discussion as it multiplies all the terms in these expressions except for  $g$  in  $EV$ . We scale  $g$  appropriately.

### 3. Analysis

The firm wants to maximize its surplus in the contract it signs with a rookie programmer. Therefore, the firm's problem is:

$$\begin{aligned} & \max_{w_c, w_o} EV(x^*) \\ \text{subject to: } & x^* \in \operatorname{argmax}_{x: 0 \leq x \leq 1} ES(x) \quad (\text{Incentive Compatibility (IC) Constraint}) \quad (4) \end{aligned}$$

$$\begin{aligned} & ES(x^*) \geq u \quad (\text{Individual Rationality (IR) Constraint}) \quad (5) \\ & w_c, w_o \geq 0 \end{aligned}$$

The *Individual Rationality* constraint, in Expression 5, expresses the requirement that the programmer get a minimum utility  $u$  from employment with the firm. Note that with the career concerns this is not just a constraint on wages and bonuses but also the value from future prospects.

The *Incentive Compatibility* constraint in Equation 4 results from the assumption that given the terms of the employment contract, the programmer picks the effort division that maximizes her personal expected surplus. Replacing 4 by its first order condition, we have:

$$\frac{dES}{dx} = p'_c w_c + p'_o w_o + (p'_t + (1-p)p'_{gb}\delta) \frac{\delta w_t}{1-\delta} = 0 \quad (IC) \quad (6)$$

In the Appendix we state and prove three lemmas that characterize the problem. In Lemma 1 we show that the effort division  $x^*$  chosen by the programmer to maximize her own surplus lies between  $x_o$  and  $x_c$ . Thus  $x^*$  has an internal value and the first order condition makes sense. If an extremely large bonus is offered for the closed project then the programmer will effectively maximize  $p_c$  which occurs at  $x_c$ . Correspondingly, if the firm offers an extremely large bonus for the open project, the worker will maximize  $p_o$ , which occurs at  $x_o$ . When no bonuses are given, Equation 6 will be satisfied when  $(p'_t + (1-p)p'_{gb}\delta) = 0$  implying that the effort choice is such that the expression  $p_t + (1-p)p'_{gb}\delta$  is maximized<sup>8</sup>. By doing this the programmer maximizes the expected payoffs due to revelation of talent which is the only payoff in absence of any incentives through bonuses. The optimal effort division for the firm depends upon the values of project successes,  $g_c$  and  $g_o$ . Thus the effort division chosen by the programmer in absence of bonuses and the one that is optimal for the firm may be quite different. The firm then provides bonuses to mitigate this discrepancy.

For the effort divisions that satisfy the incentive compatibility condition, i.e., for  $x_o < x < x_c$ , we note that  $p_o$  is decreasing and  $p_c$  is increasing in  $x$ . As a result, increasing  $w_c$  increases the effort in the closed project while increasing  $w_o$  decreases effort in the closed project. This is proven in Lemma 2.

As a last observation we note in Lemma 3 that when the Individual Rationality constraint is not binding then the firm will not offer a positive bonus for both the open and the closed source projects simultaneously in a particular contract. The intuition for this result follows from the fact that  $w_c$  and  $w_o$  have opposite effects on the effort division chosen by the programmer and having them both positive adds to the expected wage but has no benefit in aligning the incentives of the programmer and the firm.

In order to characterize the optimal bonus choices by the firm, we define  $MR$  as the marginal revenue for the firm as a function of  $x$ .

$$MR = g_c p'_c + g_o p'_o + (1-p)\delta w_t p'_{gb} \quad (7)$$

Proposition 1 lays down the optimal choice of bonuses in terms of the marginal value calculated at the effort division chosen by the programmer when the Individual Rationality constraint is trivially satisfied, for example when  $u = 0$ .

<sup>8</sup> Note that  $p_t + (1-p)p'_{gb}\delta = p_r(p + (1-p)\delta)$ . Hence the expression is concave and has a unique internal maximum due to properties of the  $p_r(z)$  function defined in Definition 1.

PROPOSITION 1. *If the Individual Rationality constraint (IR) is not binding, then the choice of bonuses that locally maximize the firm's value is reflected in the following menu:*

I.  $w_c^* = 0$  and  $w_o^* = \frac{MR^* + A^*}{p_o^* - p_o^* p_{gb}''^* / p_o^*}$  where  $x^*$  is a solution to  $\frac{MR^* + A^*}{p_o^* - p_o^* p_{gb}''^* / p_o^*} = -\frac{(p_t^* + (1-p)p_{gb}^* \delta) \frac{\delta w_t}{1-\delta}}{p_o^*}$  iff  $MR^* < -A^*$ .

II.  $w_c^* = \frac{MR^* - B^*}{p_c^* - p_c^* p_{gb}''^* / p_c^*}$  and  $w_o^* = 0$  where  $x^*$  is a solution to  $\frac{MR^* - B^*}{p_c^* - p_c^* p_{gb}''^* / p_c^*} = -\frac{(p_t^* + (1-p)p_{gb}^* \delta) \frac{\delta w_t}{1-\delta}}{p_c^*}$  iff  $B^* < MR^*$ .

III.  $w_c^* = 0$  and  $w_o^* = 0$  when  $x^*$  is a solution to  $p_t^* + (1-p)p_{gb}^* \delta = 0$  iff  $-A^* \leq MR^* \leq B^*$ .

where

$$\begin{aligned} A &= \frac{p_o}{p_o'} \frac{\delta}{1-\delta} w_t (p_t'' + (1-p)\delta p_{gb}'') \\ B &= -\frac{p_c}{p_c'} \frac{\delta}{1-\delta} w_t (p_t'' + (1-p)\delta p_{gb}'') \\ MR &= g_c p_c' + g_o p_o' + (1-p)\delta w_t p_{gb}' \end{aligned}$$

and the starred terms indicate that the corresponding quantity is evaluated at  $x^*$ , the optimal effort division chosen by the programmer.

The proof of Proposition 1 is presented in the Appendix. Since the cases depicted in this proposition represent local maximums, it may be possible for more than one case to be simultaneously satisfied. In such situations, the firm will pick the solution that provides the maximum *EV*, i.e. the one that is globally optimal. The main points conveyed through Proposition 1 are that the firm will either pay no bonuses or pay a bonus for only one of the projects. Further, positive bonus for the closed project is likely to be picked for large values of  $MR^*$  and for the open project for small values of  $MR^*$ . No bonuses are likely when  $MR^*$  has intermediate values. This is illustrated in Figure 3. To get this figure, we use the probability functions exhibited in Example 1 and numerically solve the firm's problem with the parameter values:  $\delta = 0.2$ ,  $d = 0.5$ ,  $e = 0.9$ ,  $g = 1$ ,  $g_o = 4$ ,  $p = 0.5$ ,  $w_t = 28$ ,  $u = 0$  and  $g_c$  is varied from 10 to 20. The corresponding optimal bonuses are plotted against  $MR^*$ . The bonus for the open project ( $w_o^*$ ) is plotted as a dashed line whereas the bonus for the closed project ( $w_c^*$ ) is plotted as a solid line. Note that for  $MR^* < 9.6$ , both  $w_o^*$  and  $w_c^*$  are zero. Once  $MR^* > 9.6$ ,  $w_c^* > 0$  while  $w_o^* = 0$ . Thus the figure illustrates cases (II) and (III) of Proposition 1. To relate the results of Proposition 1 to the primitive parameters, we use Figure 4. Here the optimal bonuses are plotted against  $g_c$  and the figure shows that with increasing  $g_c$ , the bonuses change the same way as with increasing  $MR^*$ , i.e., at low values of  $g_c$  both bonuses are zero, but with increasing  $g_c$  the bonus for only the closed project becomes positive. In other numerical experiments, we observe that  $w_t$  behaves similar to  $g_c$  while  $g_o$  and  $p$  have an opposite behavior.

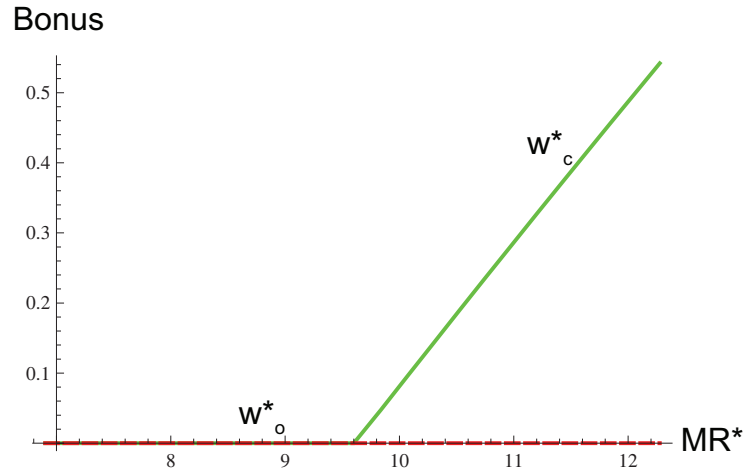


Figure 3 Illustrating Proposition 1: Bonuses with IR constraint not binding

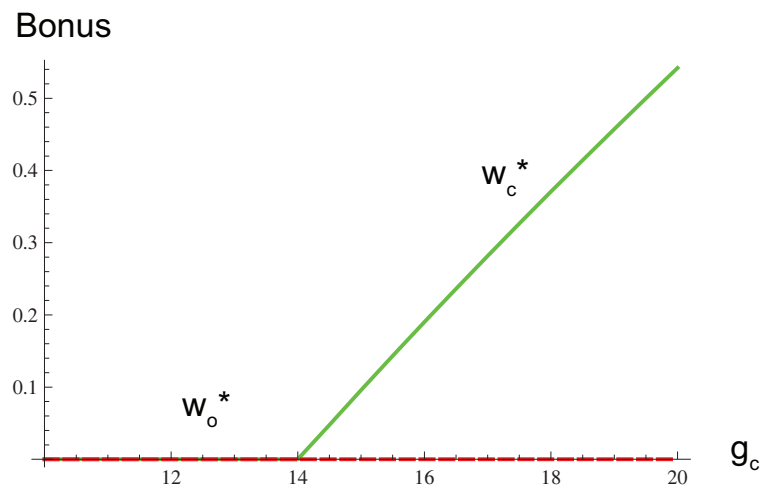


Figure 4 Illustrating Proposition 1: Bonuses with IR constraint not binding

We next want to explore how the bonuses are changed when the programmer has a minimum utility that must be met for her to stay employed, i.e., if the Individual Rationality constraint applies. In particular, our interest is to establish the contract characteristics when the firm pays bonuses for both projects simultaneously. Intuitively, this will happen when paying a bonus for only one project to meet the *IR* constraint distorts the programmer's choice of effort division too far away from the firm's optimal choice. Our next result describes this type of solution.

**PROPOSITION 2.** *An optimal solution  $(x^*, w_c^*, w_o^*)$ , where  $w_c^* > 0, w_o^* > 0$ , when the Individual Rationality Constraint (*IR*) is binding is given as follows:*

1.  $x^*$  maximizes total value  $EV_t = g + p_c g_c + p_o g_o + p_o \frac{\delta w_t}{1-\delta} + p_{gb} \frac{\delta w_t}{1-\delta}$ .
2. Using  $x^*$ ,  $w_c^*$  and  $w_o^*$  are obtained by simultaneously solving  $ES(x^*) = u$  and  $\frac{dES}{dx} \Big|_{x=x^*} = 0$ .

The proof is presented in the Appendix. Proposition 2 establishes that the solution with bonuses for both projects is also socially optimal since this maximizes the total value created for the firm and the programmer. We refer to this total value as  $EV_t$  where

$$EV_t = g + p_c g_c + p_o g_o + p_o \frac{\delta w_t}{1-\delta} + p_{gb} \frac{\delta w_t}{1-\delta}.$$

This is surprising since one might think that offering both types of bonuses must be inefficient as they have opposite effects on the choice of effort division (Lemma 2). The key intuition behind this result is as follows. Suppose the firm implements an effort division  $x^*$ . This results in firm value  $EV(x^*)$  which can be expressed as  $EV_t(x^*) - ES(x^*)$  by rewriting Equation 3. Note that  $ES(x^*) = u$  since the *IR* is binding. Thus the firm value can be maximized by implementing an  $x^*$  that maximizes  $EV_t(x)$ .

Finally, it is easily shown that  $EV_t'' < 0$  and  $EV_t' \Big|_{x=x_o} > 0$  while  $EV_t' \Big|_{x=x_c} < 0$  implying that the appropriate  $x^*$  to maximize total value  $EV_t$  is such that  $x_o < x^* < x_c$ . From Lemma 1, we know that such an  $x^*$  can always be implemented with an appropriate choice of bonuses.

Having identified the solution with positive bonuses for both projects (Proposition 2), we now characterize when this particular solution will be picked by the firm, if at all.

**PROPOSITION 3.** *When the unconstrained solution specified in Proposition 1 is not feasible due to the Individual Rationality (IR) constraint, the firm's choice of bonuses that locally maximizes its value is reflected in the following menu:*

I.  $w_c^* = 0$  and  $w_o^*$  and  $x^*$  are obtained by simultaneously solving  $ES(x^*) = u$  and  $\frac{dES}{dx} \Big|_{x=x^*} = 0$  iff

$$MR^* < -\frac{(1-\frac{p_c^*}{p_o^*})A^*}{1-\frac{p_c^*}{p_o^*}} - \frac{u-Q^*}{p_o^*} \left( p_o'^* - \frac{(p_o^*-p_c^*)p_o''^*}{p_o'^*-p_c'^*} \right).$$

II.  $w_o^* = 0$  and  $w_c^*$  and  $x^*$  are obtained by simultaneously solving  $ES(x^*) = u$  and  $\frac{dES}{dx} \Big|_{x=x^*} = 0$  iff

$$MR^* > -\frac{(1-\frac{p_o^*}{p_c^*})B^*}{1-\frac{p_o^*}{p_c^*}} - \frac{u-Q^*}{p_c^*} \left( p_c'^* - \frac{(p_o^*-p_c^*)p_c''^*}{p_o'^*-p_c'^*} \right).$$

III. The solution specified in Proposition 2 iff  $(MR^* - p_c'^* w_c^* - p_o'^* w_o^*) \frac{p_o'^*}{p_c''^* w_c^* + p_o''^* w_o^* + Q''^*} - p_o^* =$

$$(MR^* - p_c'^* w_c^* - p_o'^* w_o^*) \frac{p_c'^*}{p_c''^* w_c^* + p_o''^* w_o^* + Q''^*} - p_c^*.$$

where  $Q = (p_t + (1-p)p_{gb}\delta) \frac{\delta w_t}{1-\delta}$  and the starred terms indicate that the corresponding quantity is evaluated at  $x^*$ , the optimal effort division chosen by the programmer.

As in Proposition 1, the cases in Proposition 3 too represent local solutions and in case of multiple solutions being possible, the firm will pick the one that is globally optimal. A comparison of the

menu of bonuses given in Propositions 1 and 3 indicates that the firm's choices are significantly different when the Individual Rationality constraint is binding vis-a-vis when this constraint does not apply. The important reason for this difference is reflected by the presence of  $u$ , explicitly or implicitly, in the conditions for the three cases in Proposition 3. Of particular interest is the situation when the firm provides positive bonuses for both projects i.e. Case (III) of Proposition 3.

To better understand the situations when such a choice is made we use Figures 5 and 6. To draw these figures, we start with a situation when  $u$  is so small that the IR constraint is not binding and hence Proposition 1 applies. Then we increase  $u$  so that the IR constraint begins to bind and thereby show the impact of changing  $u$ . The parameter values used for drawing Figure 5 are  $\delta = 0.2$ ,  $d = 0.5$ ,  $e = 0.9$ ,  $g = 1$ ,  $g_c = 12$ ,  $g_o = 14$ ,  $p = 0.5$ ,  $w_t = 28$  and  $u$  is varied from 4 to 6.2. The firm's problem is numerically solved to determine the optimal bonuses using the probability functions specified in Example 1. The region to the left of the dark vertical line ( $u < 4.2$ ), is the one where  $u$  is so small that the IR constraint does not bind. The figure shows that Case (III) of Proposition 1 applies and  $w_c^* = 0$  and  $w_o^* = 0$ . As  $u$  increases beyond 4.2, the IR constraint binds. Lemma 4 proves that increase in any one of the bonuses increases the expected surplus,  $ES$ , of the rookie programmer. Figure 5 shows that the firm first meets the IR constraint by raising only one of the bonuses ( $w_c^*$ ). Increasing  $w_c^*$  beyond the optimal level to meet the IR constraint makes  $dEV/dw_c$  increasingly more negative. This eventually makes  $dEV/dw_o > dEV/dw_c$  (see proof of Proposition 3 in Appendix) and the firm finds that it is now attractive to raise  $w_o$  as well. In Figure 5 this happens when  $u$  increases beyond 5.4. The firm then provides  $w_c^* > 0$  and  $w_o^* > 0$  to meet the IR constraint. Figure 6 shows another instance of how increase in  $u$  impacts bonus choices. The difference from Figure 5 is that initially the firm picks  $w_o^*$  to meet the constraint and increases  $w_c^*$  only later. The reason for the difference is that in drawing Figure 6 the parameter  $g_o$  is increased to 25 and  $g_c$  is reduced to 5 while others are held fixed. This is useful to showcase that in Figure 5 cases (II) and (III) of Proposition 3 and in Figure 6 cases (I) and (III) of Proposition 3 apply.

We now comment on the social efficiency of the various solutions in Propositions 1 and 3. The point  $x_2$  in the figure corresponds to the effort division that maximizes social efficiency. Proposition 2 establishes that this is the effort division that is picked by the programmer when  $w_c^* > 0$ ,  $w_o^* > 0$  and the IR constraint is binding. In Lemma 5, presented in the Appendix, we show that  $dEV_i/dx > 0$  when  $w_c^* > 0$ ,  $w_o^* = 0$  and the IR constraint is not binding. Hence, in this case the bonuses offered by the firm induce the programmer to pick an effort division that is *too small* for social efficiency. This is illustrated in Figure 7 by point  $x_1$ . When the IR constraint binds, but  $w_c^* > 0$  and  $w_o^* = 0$ , the bonus for the closed project must be raised to meet the IR constraint. Hence, in accordance

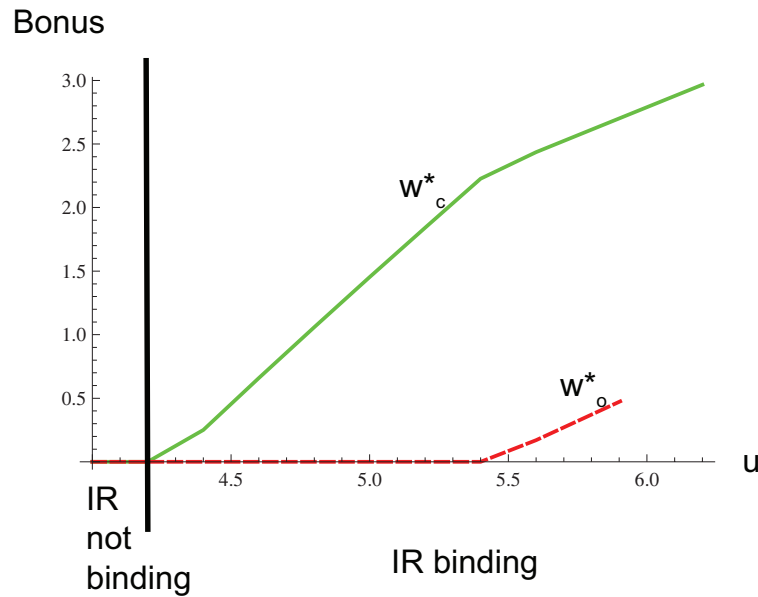


Figure 5 Illustrating Proposition 3: Bonuses with increasing  $u$

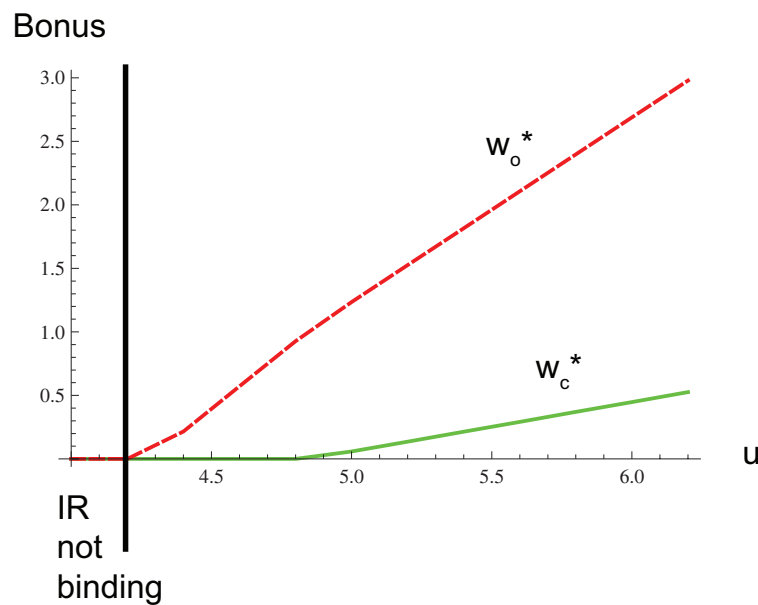
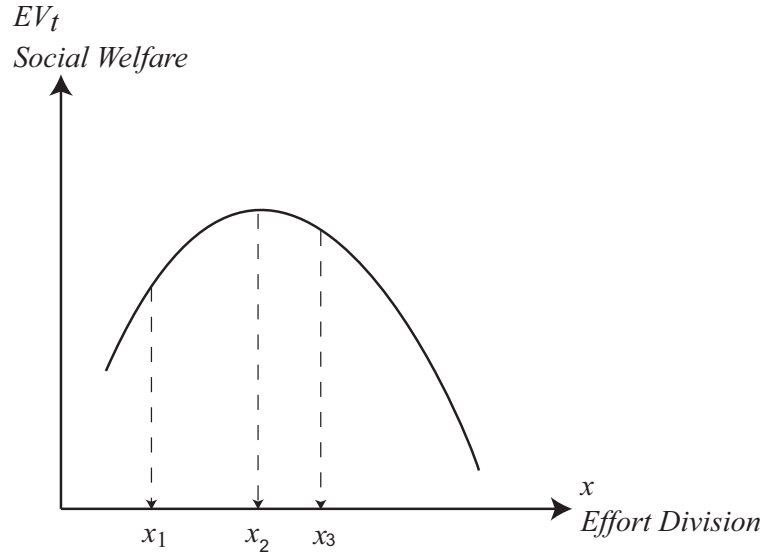


Figure 6 Another illustration of Proposition 3: Bonuses with increasing  $u$

with Lemma 2, the equilibrium effort level increases. This reduces the discrepancy from the socially optimal effort level, at least initially. If the IR constraint is not binding and  $w_o^* > 0$  and  $w_c^* = 0$ , then, as shown in Lemma 6 in the Appendix, the effort choice in equilibrium is *too high* for social efficiency ( $dEV_i/dx < 0$ ). This is illustrated in Figure 7 by point  $x_3$ . As before, when the IR constraint binds, the bonus for the open project may be increased and this may reduce the



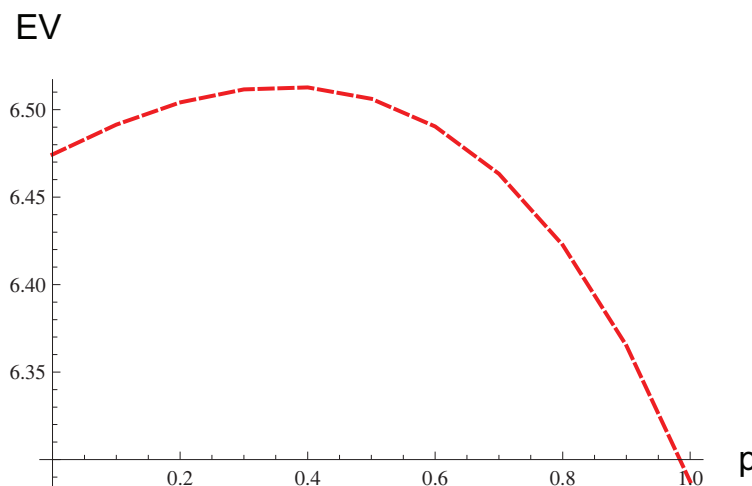
**Figure 7** Social efficiency

discrepancy from the socially optimum effort.

#### 4. Impact of Degree of Openness of the Proprietary Project

The open source project is superior to the closed source project for the programmer in that it more clearly reveals the talent of the programmer. In absence of any direct incentives offered by the firm, this built in career incentive dictates the effort division of the programmer between the open and closed source projects. In the preceding analysis we showed that such effort division may not be optimal for the firm and it may give bonuses to induce a different effort division.

Another way in which the firm could possibly induce a favorable effort division is by changing the degree of openness of the proprietary project. As the proprietary project is made more open, success in this project is more likely to convey to the labor market that the programmer is talented. Thus the programmer will pick an effort division that is more in favor of the proprietary project. Increasing the degree of openness of the proprietary project however comes at a cost to the firm because this reduces the chances that the firm will have private information that the programmer is talented. Hence the benefits from employing a talented programmer at a low wage cost would reduce. Increasing the degree of openness to induce higher effort in the closed project therefore comes with its own cost-benefit tradeoff. This indicates that some intermediate level of openness of the proprietary project may be optimal for maximizing the firm's value, as Figure 8 shows. This figure is drawn using the parameter values  $\delta = 0.2$ ,  $d = 0.5$ ,  $e = 0.9$ ,  $g = 1$ ,  $g_c = 4$ ,  $g_o = 6$ ,  $u = 0$  and  $w_t = 28$ . Recall that the programmer will exert effort  $x_r(p + (1 - p)\delta)$  to maximize the probability  $p_r(p + (1 - p)\delta)$  in absence of any bonuses. If it is optimal for the firm to implement an effort in



**Figure 8** Impact of changing  $p$  with bonus for only one project

the closed project greater than  $x_r(p + (1 - p)\delta)$ , then it has the two options of either increasing  $w_c$  or increasing  $p$ . It is in such cases that increasing the degree of openness turns out to be a useful. If, on the other hand, the optimal decision for the firm is to implement an effort lesser than  $x_r(p + (1 - p)\delta)$  in the closed project, then it should either reduce the degree of openness of the closed project, or increase  $w_o$ .

If the  $IR$  constraint is binding, the optimal decision for the firm changes since it now needs to ensure a minimum payoff to the programmer. If it is paying a bonus in only one project to ascertain the minimum payoff, the flexibility of adjusting the degree of openness may allow the firm to do this at a lower cost and may therefore improve the firm's value. If, however, the firm is paying bonuses on both projects, it picks the bonuses to implement an effort that maximizes the total surplus  $EV_t$ . The firm value  $EV = EV_t - u$ , therefore, remains unchanged even if the degree of openness of the closed project can be adjusted. This happens because as the degree of openness of the closed project increases, the programmer gets additional future benefits due to increase in chances of revelation of talent. Consequently, the firm implements the socially optimal effort by lowering bonus payments. However, the savings in cost from bonus payments are exactly compensated for by the increase in cost to the firm due to increase in  $p$ . Hence the net cost to the firm and the firm value remain the same.

## 5. Concluding Remarks

### 5.1. Discussion on Model Assumptions and Limitations

An artifact of our model is the assumption that only talented programmers can produce a successful outcome which can then be construed as an informational signal of talent. If even untalented programmers can sometimes create a successful outcome, the major impact on our model will be

that output created by rookies in the first period of employment will not be sufficient to conclude whether they are talented. Now information on the output created over several periods will be needed to make a judgment of the programmers' talent. Hence the premium for being a talented programmer, reflected in the market wage  $w_t$ , will be available only after several periods. Clearly, the future payoffs to signaling talent are reduced. The choice of effort division is made by the programmers' to maximize their combined payoff from the future payoffs and the expected bonus payment in the current period. Since the future becomes less valuable with the relaxing of this assumption, the firm may be able to influence the programmers' choice of effort division by paying a lower bonus and therefore results in a lower expected surplus for the programmers. However, this makes the IR constraint bind more quickly. The firm must then raise the bonus to meet the IR constraint until paying a bonus for just one project to meet the IR constraint results in a highly distorted effort division by the programmer. The firm will then have to resort to paying bonuses for both projects, just as in the current model. Thus our key insights are expected to be robust to relaxing this assumption.

A limitation is that we have only focused on learning by doing and not on learning through classroom training. Thus our model assumes that the impact of learning by doing is in addition to the learning that is possible through classroom training. Support for the hypothesis that learning by doing provides unique learning opportunities that cannot be duplicated through learning by training is available through various sources such as the European Commission sponsored report on open source software, available at <http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf>. This report also points out that many firms value the learning benefits that their programmers get by working on open source projects. Hence they explicitly or implicitly allow their programmers to work in open source projects during work time.

If learning by training were modeled, one fundamental difference from learning by doing through open source is the moral hazard issue. While classroom training time can be explicitly controlled by the firm, there is little control on the programmers' mindshare when they are allowed to simultaneously work on open and closed source projects. Consequently, firms can control the training time directly without providing any bonuses to programmers. Another major difference is that classroom training may not be as good a way to convey one's talent to the market as a good outcome in a live open source project. Thus classroom training may have less career incentive connotations. Finally, classroom training comes at a cost while doing open source is free and may even have strategic value for the firm (which we capture through  $g_o$ ). Due to these differences, learning by training and learning by doing will have different implications from the firm's and programmers' perspectives.

In particular, no bonus payments are necessary to induce programmers to learn through training as the firm can explicitly control programmers' participation in these activities.

Assumptions 1 and 2 are central to our model and capture the impact of effort and learning by doing effects. Due to the tension in the effort and learning by doing, the marginal probabilities of success are assumed to be concave with internal maxima. If, however, the learning by doing effects are not strong enough, the concavity of the marginal probabilities and the internal maxima will disappear and our results will not apply. As an outcome of this assumption, the probabilities  $p_r(z)$  also turn out to be concave. Assumption 2 makes the point that the impact of learning by doing is not too large. Relaxing these assumptions may induce a programmer to choose effort  $x^* = 0$ , or  $x^* = 1$ . Since we are not interested in corner solutions, our assumptions help eliminate these uninteresting situations and allow us to focus on situations where the learning effects are significant but at the same time they are not too large so as to have an overriding influence.

## 5.2. Discussion of the Setting and Results

It has been observed that many programmers hired by software development firms contribute to open source software projects in addition to working at the projects for which they have been hired. Previous researchers have addressed this puzzle by pointing out that these programmers may derive some economic value from learning and career progression opportunities since good contributions in open source projects help convince the market that the concerned programmers are talented and deserve better compensation in future.

When the programmers work on open source projects, they use up their time which they could have potentially invested in the proprietary projects for the hiring firms. This is not always harmful to the hiring firm since they may have several sources of value from open source projects. One such situation is the “Eclipse” open source development platform supported by IBM. IBM's programmers work on Eclipse while simultaneously working on the development of software tools complementary to the Eclipse platform. These software tools are commercially sold by IBM. Clearly, IBM's business depends on the popularity and success of the Eclipse project. In other words, IBM derives a strategic value from the success of the Eclipse platform. In addition to such strategic value, when a firm's programmers work on open source projects they may learn useful skills which improves their success rate on the firm's proprietary project and vice-versa. Again, the Eclipse platform and its complementary software are a good example. There is clearly a lot of learning benefit because both projects are closely related. One can therefore surmise that both the firm and the programmers have definite incentives to succeed in both open and closed projects, but these incentives may not be aligned (e.g. programmer may have a higher incentive to do well

in the open project than the firm). Due to this situation, the firm may enter into performance based contracts with programmers to align their incentives with its own. We find that when the marginal revenue of effort for the closed project is high, the firm pays a bonus for this project. When this marginal revenue is low, the firm pays a bonus for the open project. For the mid-range of marginal value the firm pays no bonuses. The programmers' then make an effort choice that maximizes the chance of exposing her talent in the labor market to get the wage premium for talented programmers in future. If however, the outside opportunities are attractive, future benefits may not be sufficient to attract the programmers to take the firm's job offer. In this case, the firm may pay a bonus for both projects. The reader may wonder if any firm has ever paid bonuses to programmers for developing open source projects. We found a real world example of that on [http://www.theserverside.com/news/thread.tss?thread\\_id=18524#77810](http://www.theserverside.com/news/thread.tss?thread_id=18524#77810). This news item (article date March 25, 2003) reports that the software firm JBoss paid bonuses to several programmers who contributed to the development of an open source application server pioneered by this firm. We further note that a socially efficient outcome is obtained in circumstances when the firm offers bonuses for both the projects. This is because with the bonuses chosen by the firm, the programmer picks an effort division that maximizes the total expected value created.

If the firm employs policies that showcase the programmers' contributions in the proprietary project in a much better way to the other market firms, the programmers will have a higher incentive to invest effort in this project. Thus making the proprietary project more open can serve as a substitute to increasing the bonus for this project. However, such a policy comes at a cost to the firm since it reduces the chances of the firm having access to the services of a talented programmer at low wage costs. The strategy employed by Firaxis Games is similar to the one we discuss here. This firm owns the rights to a very famous game named *Civilization*. It lists the names of important developers of this game on its web site. Adobe Inc. also employs a similar policy. When the Adobe Photoshop CS2 version 9.0 loads, the names of the important contributors are listed. This kind of policy is likely to be effective only when the outside market opportunities are not too attractive. When this is not the case, increasing the degree of openness for the closed project is unlikely to yield any benefit.

In summary then, this paper analyzes a new business model based on the now well understood economic incentives of programmers to contribute to open source software. The contribution is particularly timely as firms today continue to learn to leverage open source effectively.

## References

- ARROW, K. J. (1962): “The Economic Implications of Learning by Doing,” *The Review of Economic Studies*, 29(3), 155–173.
- BECKER, G. S. (1962): “Investment in Human Capital: A Theoretical Analysis,” *Journal of Political Economy*, 40, 9–49.
- BLAUG, M. (1976): “The Empirical Status of Human Capital Theory: A Slightly Jaundiced Survey,” *Journal of Economic Literature*, 16, 827–855.
- BOH, W. F., S. A. SLAUGHTER, AND J. A. ESPINOSA (2007): “Learning from Experience in Software Development: A Multilevel Analysis,” *Management Science*, 53(8), 1315–1331.
- BRESHNAHAN, T. F., E. BRYNJOLFSSON, AND L. M. HITT (2002): “Information Technology, Workplace Reorganization, and the Demand for Skilled Labor: Firm-Level Evidence,” *Quarterly Journal of Economics*, pp. 339–376.
- CAROLI, E., AND J. V. REENEN (2001): “Skill Based Organizational Change? Evidence From a Panel of British and French Establishments,” *Quarterly Journal of Economics*, pp. 1449–1492.
- CARTENSEN, V. (2002): “The From-Tayloristic-to-Holistic-Organization Model From an Empirical Perspective,” *Discussion paper No. 256, University of Hanover*.
- FINK, M. (2002): *The Business and Economics of Linux and Open Source*. Prentice Hall PTR, Indianapolis.
- GIBBONS, R., AND K. J. MURPHY (1992): “Optimal Incentive Contracts in the Presence of Career Concerns: Theory and Evidence,” *Journal of Political Economy*, 100(3), 468–504.
- GOLEMAN, D. (1998): *Working with emotional intelligence*. Bantam Books, New York, NY.
- HANN, I. H., J. ROBERTS, S. SLAUGHTER, AND R. FIELDING (2006): “Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects,” *Management Science*, 52(7).
- HARS, A., AND S. OU (2002): “Working For Free? Motivations for Participating in Open Source Projects,” *International Journal of Electronic Commerce*, 6(3).
- HERTEL, G., S. NIEDNER, AND S. HERRMANN (2003): “Motivation of Software Developers in Open Source Projects: An Internet Based Survey of Contributors to the Linux Kernel,” *Research Policy*, 32, 1159–1177.
- HOLMSTROM, B. (1999): “Managerial Incentive Problems: A Dynamic Perspective,” *Review of Economic Studies*, 66(1), 169–182.
- JOHNSON, J. P. (2002): “Open Source Software: Private Provision of a Public Good,” *Journal of Economics and Management Strategy*, 11(4), 637–662.
- KILLINGSWORTH, M. R. (1982): ““Learning by doing” and “Investment in Training”: A Synthesis of Two “Rival” Models of the Life Cycle,” *The Review of Economic Studies*, 49(2), 263–271.

- KIRSCH, L. J. (1996): “The Management of Complex Tasks in Organizations: Controlling the Systems Development Process,” *Organization Science*, 7, 1–21.
- LAKHANI, K. R., AND E. VON HIPPEL (2003): “How Open Source Software Works: “Free” User-to-User Assistance,” *Research Policy*, 32, 923–943.
- LAKHANI, K. R., AND R. G. WOLF (2005): *Perspectives on Free and Open Source Software*. MIT Press, Cambridge, Massachusetts, USA.
- LAL, R., AND V. SRINIVASAN (1993): “Compensation Plans for Single and Multi-Product Salesforces: An Application of the Holmstrom-Milgrom Model,” *Management Science*, 39(7), 777–793.
- LERNER, J., AND J. TIROLE (2002): “Some Simple Economics of Open Source,” *Journal of Industrial Economics*, 50(2), 197–234.
- LINDBECK, A., AND D. J. SNOWER (1996): “Reorganization of Firms and Labor-Market Inequality,” *The American Economic Review*, 86(2), 315–321.
- LINDBECK, A., AND D. J. SNOWER (2000): “Multitask Learning and the Reorganization of Work: From Tayloristic to Holistic Organization,” *Journal of Labor Economics*, 18(3), 353–376.
- MENON, T., AND J. PFEFFER (2003): “Valuing Internal vs. External Knowledge: Explaining the Preference for Outsiders,” *Management Science*, 49(4), 497–513.
- MILBOURN, T. T., R. R. SHOCKLEY, AND A. V. THAKOR (2001): “Managerial Career Concerns and Investments in Information,” *RAND Journal of Economics*, 32(2), 334–351.
- MOODY, G. (2002): *Rebel Code: Inside Linux and the Open Source Revolution*. Perseus Press, New York, NY.
- NEAL, D. (1995): “Industry Specific Human Capital: Evidence from Displaced Workers,” *Journal of Labor Economics*, 13(4), 653–677.
- New York Times Magazine (Oct 10, 1999): “The Search Engine,” .
- PAUTREL, X. (2004): “A Note on Multitask Learning and the Reorganization of Work,” *Economics Bulletin*, 10(5), 1–6.
- RAYMOND, E. S. (2001): *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O’Reilly Publications.
- REAGANS, R., L. ARGOTE, AND D. BROOKS (2005): “Individual Experience and Experience Working Together: Predicting Learning Rates from Knowing Who Knows What and Knowing How to Work Together,” *Management Science*, 51(6), 869–881.
- ROSSI, C., AND A. BONACCORSI (2006): “Intrinsic Motivations and Profit-Oriented Firms in OSS,” *The Economics of Open Source Software Development*.
- SCHILLING, M. A., P. VIDAL, R. E. POLYHART, AND A. MARANGONI (2003): “Learning by Doing *Something* Else: Variation, Relatedness, and the Learning Curve,” *Management Science*, 49(1), 39–56.

United Nations University UNU-MERIT (2006): *Study on the Economic Impact of Open Source Software on Innovation and the Competitiveness of the Information and Communication Technologies (ICT) Sector in the EU*. Netherlands.

VON KROGH, G., S. SPAETH, AND K. R. LAKHANI (2003): “Community, Joining, and Specialization in Open Source Software Innovation: A Case Study,” *Research Policy*, 32(7).

WAYNER, P. (2000): *Free For All: How Linux and the Free Software Movement Undercut the High-Tech Titans*. HarperBusiness, New York.

WEISS, A. (1995): “Human Capital Vs Signaling Explanation of Wages,” *Journal of Economic Perspectives*, 9, 133–154.

## Appendix

LEMMA 1. *Let  $x_c$  and  $x_o$  be the maxima over  $[0, 1]$  of  $p_c$  and  $p_o$ , respectively. Only  $x^*$  such that  $x_o \leq x^* \leq x_c$  can satisfy the Incentive Compatibility (IC) constraint.*

*F* first note that Assumption 1 says that  $0 < x_o < x_c < 1$ . We note that all terms of  $dES/dx$  are weakly positive if  $x < x_o$  and so the programmer would prefer a larger  $x$ . Hence  $x^* \geq x_o$ .

Turning to  $x_c$ , we note that all terms of  $dES/dx$  are weakly negative if  $x > x_c$  and so the programmer would prefer a smaller  $x$ . Hence  $x^* \leq x_c$ .

LEMMA 2. *If  $x^*$  satisfies the Incentive Compatibility constraint then  $\frac{\partial x^*}{\partial w_c} \geq 0$  and  $\frac{\partial x^*}{\partial w_o} \leq 0$ .*

*B*y using the Implicit Function Theorem on Equation (6) ,

$$\frac{\partial x^*}{\partial w_c} = -ES_{xw_c}/ES_{xx} = -\frac{p'_c}{p'_c w_c + p'_o w_o + (p'_t + (1-p)\delta p''_{gb})\delta w_t / (1-\delta)} \quad (8)$$

Note that  $(p'_t + (1-p)\delta p''_{gb}) = p''_r(p + (1-p)\delta)$ , and from Definition 1 we know that  $p''_r(z) < 0$ . Hence, the denominator is negative by the strict concavity of the probability functions. The numerator,  $p'_c$ , is positive in the interval  $(x_o, x_c)$ . This proves that  $\frac{\partial x^*}{\partial w_c} \geq 0$ .

The proof of  $\frac{\partial x^*}{\partial w_o} \leq 0$  is similar and is omitted here.

LEMMA 3. *If the Individual Rationality constraint (IR) is not binding, then the firm sets either  $w_c^* > 0$  or  $w_o^* > 0$ , but not both.*

*L*et  $(w_c^*, w_o^*)$  be any solution to the firm’s problem. Let  $x^*$  be the effort level on the closed project chosen by the worker with these bonuses. Consider a problem of minimizing the cost of getting the worker to implement  $x^*$ . Clearly,  $(w_c^*, w_o^*)$  must solve this problem too or else these bonuses do not solve the firm’s problem as we have assumed. We will now use this simplified problem to characterize the optimal bonuses.

At a given  $x^*$ , the problem of finding a least cost set of bonuses to implement  $x^*$  is a linear programming problem with two decision variables,  $w_c$  and  $w_o$ , and one constraint – the IC constraint (as the IR constraint is not binding). Since there is only one constraint, only one of the variables,  $w_c$  or  $w_o$ , will enter the basis. Therefore, either  $w_c^* > 0$  or  $w_o^* > 0$ , but not both.

PROPOSITION 1. *If the Individual Rationality constraint (IR) is not binding, then the choice of bonuses that locally maximize the firm's value is reflected in the following menu:*

$$I. w_c^* = 0 \text{ and } w_o^* = \frac{MR^* + A^*}{p_o'^* - p_o^* p_o''^* / p_o'^*} \text{ where } x^* \text{ is a solution to } \frac{MR^* + A^*}{p_o'^* - p_o^* p_o''^* / p_o'^*} = - \frac{(p_t'^* + (1-p)p_{gb}'^* \delta) \frac{\delta w_t}{1-\delta}}{p_o'^*} \text{ iff } MR^* < -A^*.$$

$$II. w_c^* = \frac{MR^* - B^*}{p_c'^* - p_c^* p_c''^* / p_c'^*} \text{ and } w_o^* = 0 \text{ where } x^* \text{ is a solution to } \frac{MR^* - B^*}{p_c'^* - p_c^* p_c''^* / p_c'^*} = - \frac{(p_t'^* + (1-p)p_{gb}'^* \delta) \frac{\delta w_t}{1-\delta}}{p_c'^*} \text{ iff } B^* < MR^*.$$

$$III. w_c^* = 0 \text{ and } w_o^* = 0 \text{ when } x^* \text{ is a solution to } p_t'^* + (1-p)p_{gb}'^* \delta = 0 \text{ iff } -A^* \leq MR^* \leq B^*.$$

where

$$A = \frac{p_o}{p_o'} \frac{\delta}{1-\delta} w_t (p_t'' + (1-p)\delta p_{gb}'')$$

$$B = -\frac{p_c}{p_c'} \frac{\delta}{1-\delta} w_t (p_t'' + (1-p)\delta p_{gb}'')$$

$$MR = g_c p_c' + g_o p_o' + (1-p)\delta w_t p_{gb}'$$

and the starred terms indicate that the corresponding quantity is evaluated at  $x^*$ , the optimal effort division chosen by the programmer.

Suppose the firm chooses  $w_c^* = 0$ . By Lemma 3 this implies that  $w_o^* \geq 0$ .

Taking  $w_c^* = 0$ , we examine the first order condition for the firm's objective function wrt  $w_o$ .

$$\left. \frac{dEV}{dw_o} \right|_{w_c^*=0, x=x^*} = (MR^* - p_o'^* w_o) \frac{dx^*}{dw_o} - p_o^* = 0$$

We substitute the value of  $\frac{dx^*}{dw_o}$ , which is obtained in a fashion similar to the way  $\frac{dx^*}{dw_c}$  was obtained in Lemma 2, in the above equation and solve for  $w_o^*$ . This gives

$$w_o^* = \frac{MR^* + A^*}{p_o'^* - p_o^* p_o''^* / p_o'^*} \quad (9)$$

where  $A^* = \frac{p_o}{p_o'} \frac{\delta}{1-\delta} w_t (p_t'' + (1-p)\delta p_{gb}'')$ . Note that the numerator of  $A^*$  is negative since  $(p_t'' + (1-p)\delta p_{gb}'') = p_r''(p + (1-p)\delta)$ , and from Definition 1 we know that  $p_r''(z) < 0$ . The denominator of  $A^*$  is negative because  $p_o'$  negative at  $x \geq x_o$ . Hence  $A^*$  is a positive quantity. Note that so is  $w_o^*$  provided  $MR^* < -A^*$  as  $w_o^*$  then has a negative numerator and denominator. This also rules out taking  $w_o^* < 0$ .

Given that  $w_o^*$  and  $w_c^*$  are chosen as above, the programmer chooses the effort  $x^*$  that maximizes her surplus  $ES$ . The corresponding first order condition is  $p'_o w_o + p'_c w_c + (p'_t + (1-p)p'_{gb}\delta) \frac{\delta w_t}{1-\delta} = 0$ . Substituting in  $w_c^*$  and  $w_o^*$  from above and simplifying, we get  $\frac{MR^* + A^*}{p_o^* - p_o^* p_o^* / p_o^*} = - \frac{(p_t^* + (1-p)p_{gb}^* \delta) \frac{\delta w_t}{1-\delta}}{p_o^*}$ , which gives us the solution for  $x^*$ .

Note that the  $w_c^*$ ,  $w_o^*$  and  $x^*$  found above form a consistent solution iff  $MR^* < -A^*$ .

Cases (II) and (III) are proved similarly.

To complete the proof we also show that for any values of  $A^*$  and  $B^*$  it must be true that  $-A^* < B^*$ . We already established that  $A^* > 0$ . Take  $B^*$ . Employing parallel logic to that used for showing  $A^* > 0$ , we see that its numerator is negative and denominator is positive. With the negative sign of the expression, we have  $B^* > 0$ . Since, both  $A^*$  and  $B^*$  are positive quantities, it must be that  $-A^* < B^*$ . Therefore, for some parameter values, the solution specified in Case (III) is the only solution.

LEMMA 4. *Increasing  $w_c$  or  $w_o$  increases programmer's expected surplus.*

Using Equation 2, we can write

$$\frac{dES}{dw_c} = \left( ES' \frac{dx}{dw_c} \right) \Big|_{x=x^*} + p_c^*$$

. Since  $ES' \Big|_{x=x^*} = 0$ , we have  $\frac{dES}{dw_c} = p_c^* > 0$ . Similarly, we can show that  $\frac{dES}{dw_o} > 0$ .

PROPOSITION 2. *An optimal solution  $(x^*, w_c^*, w_o^*)$ , where  $w_c^* > 0, w_o^* > 0$ , when the Individual Rationality Constraint (IR) is binding is given as follows:*

1.  $x^*$  maximizes total value  $EV_t = g + p_c g_c + p_o g_o + p_o \frac{\delta w_t}{1-\delta} + p_{gb} \frac{\delta w_t}{1-\delta}$ .
2. Using  $x^*$ ,  $w_c^*$  and  $w_o^*$  are obtained by simultaneously solving  $ES(x^*) = u$  and  $\frac{dES}{dx} \Big|_{x=x^*} = 0$ .

Equation (3) can be rewritten as  $EV(x) = EV_t(x) - ES(x)$ , where  $EV_t(x) = g + p_c g_c + p_o g_o + p_o \frac{\delta w_t}{1-\delta} + p_{gb} \frac{\delta w_t}{1-\delta}$ . The Lagrangian of the firm's problem is:

$$L = EV_t(x^*) + (\lambda - 1)ES(x^*) + \mu w_c + \nu w_o - \lambda u,$$

where  $x^*$  has been substituted in from the Incentive Compatibility constraint (IC).  $\lambda$ ,  $\mu$  and  $\nu$  are the Lagrange multipliers of the Individual Rationality constraint (IR), and the non-negativity constraints on  $w_c$  and  $w_o$  respectively. Dropping the parenthesis in the expression for the Lagrangian and taking its derivatives with respect to  $w_c$  and  $w_o$ , we get

$$\frac{dL}{dw_c} = \frac{dEV_t}{dw_c} + (\lambda - 1)p_c + \mu = 0 \tag{10}$$

$$\frac{dL}{dw_o} = \frac{dEV_t}{dw_o} + (\lambda - 1)p_o + \nu = 0 \quad (11)$$

If  $w_o, w_c > 0$ , then  $\mu = 0$  and  $\nu = 0$ . Further,  $\frac{dEV_t}{dw_c} = (p'_c g_c + p'_o g_o + p'_o \frac{\delta w_t}{1-\delta} + p'_{gb} \frac{\delta w_t}{1-\delta}) \frac{dx^*}{dw_c}$  and  $\frac{dEV_t}{dw_o} = (p'_c g_c + p'_o g_o + p'_o \frac{\delta w_t}{1-\delta} + p'_{gb} \frac{\delta w_t}{1-\delta}) \frac{dx^*}{dw_o}$ . Using Lemma 2,  $\frac{dEV_t}{dw_c}$  and  $\frac{dEV_t}{dw_o}$  must be of opposite signs. Hence, for the equations (10) and (11) to be consistent, it must be that  $\lambda = 1$ ,  $\frac{dEV_t}{dw_c} = 0$  and  $\frac{dEV_t}{dw_o} = 0$ .  $\lambda > 0$  implies that the Individual Rationality constraint (IR) is binding.

$\frac{dEV_t}{dw_c} = 0$  and  $\frac{dEV_t}{dw_o} = 0$  imply that  $p'_c g_c + p'_o g_o + p'_o \frac{\delta w_t}{1-\delta} + p'_{gb} \frac{\delta w_t}{1-\delta} = 0$ . We can determine the firm's choice of  $x$  to be implemented by solving this last equation. Notice that this equation also implies maximization of  $EV_t = g + p_c g_c + p_o g_o + p_o \frac{\delta w_t}{1-\delta} + p_{gb} \frac{\delta w_t}{1-\delta}$  wrt  $x$ .

Once  $x^*$  is determined, we solve (IR) and (IC) simultaneously at  $x^*$  to find the optimal bonus wages  $w_c^*$  and  $w_o^*$ . All the probabilities are evaluated at  $x^*$  and hence these two equations are linear. Therefore, using Cramer's Rule we get:

$$w_c^* = \frac{\begin{vmatrix} u - p_t \frac{\delta w_t}{1-\delta} - (1-p)p_{gb} \frac{\delta^2 w_t}{1-\delta} & p_o \\ -p'_t \frac{\delta w_t}{1-\delta} - (1-p)p'_{gb} \frac{\delta^2 w_t}{1-\delta} & p'_o \end{vmatrix}}{\begin{vmatrix} p_c & p_o \\ p'_c & p'_o \end{vmatrix}} \quad w_o^* = \frac{\begin{vmatrix} p_c & u - p_t \frac{\delta w_t}{1-\delta} - (1-p)p_{gb} \frac{\delta^2 w_t}{1-\delta} \\ p'_c & -p'_t \frac{\delta w_t}{1-\delta} - (1-p)p'_{gb} \frac{\delta^2 w_t}{1-\delta} \end{vmatrix}}{\begin{vmatrix} p_c & p_o \\ p'_c & p'_o \end{vmatrix}} \quad (12)$$

PROPOSITION 3. *When the unconstrained solution specified in Proposition 1 is not feasible due to the Individual Rationality (IR) constraint, the firm's choice of bonuses that locally maximizes its value is reflected in the following menu:*

I.  $w_c^* = 0$  and  $w_o^*$  and  $x^*$  are obtained by simultaneously solving  $ES(x^*) = u$  and  $\left. \frac{dES}{dx} \right|_{x=x^*} = 0$  iff

$$MR^* < -\frac{(1-\frac{p_c^*}{p_o^*})A^*}{1-\frac{p_c^*}{p_o^*}} - \frac{u-Q^*}{p_o^*} \left( p_o^* - \frac{(p_o^*-p_c^*)p_o^*}{p_o^*-p_c^*} \right).$$

II.  $w_o^* = 0$  and  $w_c^*$  and  $x^*$  are obtained by simultaneously solving  $ES(x^*) = u$  and  $\left. \frac{dES}{dx} \right|_{x=x^*} = 0$  iff

$$MR^* > -\frac{(1-\frac{p_o^*}{p_c^*})B^*}{1-\frac{p_o^*}{p_c^*}} - \frac{u-Q^*}{p_c^*} \left( p_c^* - \frac{(p_o^*-p_c^*)p_c^*}{p_o^*-p_c^*} \right).$$

III. The solution specified in Proposition 2 iff  $(MR^* - p'_c w_c^* - p'_o w_o^*) \frac{p_o^*}{p_c^* w_c^* + p_o^* w_o^* + Q^{**}} - p_o^* = (MR^* - p'_c w_c^* - p'_o w_o^*) \frac{p_c^*}{p_c^* w_c^* + p_o^* w_o^* + Q^{**}} - p_c^*$ .

where  $Q = (p_t + (1-p)p_{gb}\delta) \frac{\delta w_t}{1-\delta}$  and the starred terms indicate that the corresponding quantity is evaluated at  $x^*$ , the optimal effort division chosen by the programmer.

When the unconstrained solution given in Proposition 1 is not feasible, either or both of  $w_c^*$  and  $w_o^*$  must be positive to ensure that  $ES = u$ . Notice that the IR constraint can always be satisfied for some values of  $w_c$  and  $w_o$ . Hence the optimization problem always has a solution and it takes one of the following forms:

I.  $w_c^* > 0$  and  $w_o^* = 0$ .

II.  $w_c^* = 0$  and  $w_o^* > 0$ .

III.  $w_c^* > 0$  and  $w_o^* > 0$ .

First, we consider the solution specified in (I). In this case,  $w_o^* = 0$ . Then  $ES(x^*) = u$  implies  $w_c^* = \frac{u-Q^*}{p_c^*} > 0$ . Substituting in the values of  $w_c^*$  and  $w_o^*$  in the First Order condition of the expected surplus  $ES$  of the programmer we obtain  $x^*$ . The optimality of the solution depends upon the slopes of  $EV$  with respect to the bonuses  $w_c$  and  $w_o$ . We have the derivatives

$$\left. \frac{dEV}{dw_o} \right|_{w_o^*=0} = \left( g_c p_c' + g_o p_o' + (1-p) p_{gb}' \delta w_t \right) \frac{dx^*}{dw_o} - p_c' w_c \frac{dx^*}{dw_o} - p_o^* \quad (13)$$

$$\left. \frac{dEV}{dw_c} \right|_{w_o^*=0} = \left( g_c p_c' + g_o p_o' + (1-p) p_{gb}' \delta w_t \right) \frac{dx^*}{dw_c} - p_c' w_c \frac{dx^*}{dw_c} - p_o^* \quad (14)$$

Combining the above two equations, (13) and (14), we get

$$\left. \frac{dEV}{dw_o} \right|_{w_o^*=0} = \left( p_c^* + \left. \frac{dEV}{dw_c} \right|_{w_o^*=0} \right) \frac{dx^*}{dw_o} / \frac{dx^*}{dw_c} - p_o^* \quad (15)$$

From Equation (8), in the proof of Lemma 2, and a similar equation for  $dx^*/dw_o$ , we note that

$$\frac{dx^*}{dw_o} / \frac{dx^*}{dw_c} = \frac{p_o'^*}{p_c'^*} \quad (16)$$

Substituting this into (15), we get

$$\left. \frac{dEV}{dw_o} \right|_{w_o^*=0} = \left( p_c^* + \left. \frac{dEV}{dw_c} \right|_{w_o^*=0} \right) \frac{p_o'^*}{p_c'^*} - p_o^*$$

The firm will choose  $w_c^* > 0$  and  $w_o^* = 0$  as assumed iff  $\left. \frac{dEV}{dw_o} \right|_{w_o^*=0} < \left. \frac{dEV}{dw_c} \right|_{w_o^*=0}$ . Since  $p_c'^* > 0$  and  $p_o'^* < 0$ ,

$$\left. \frac{dEV}{dw_o} \right|_{w_o^*=0} < \left. \frac{dEV}{dw_c} \right|_{w_o^*=0} \quad \text{iff} \quad \left. \frac{dEV}{dw_c} \right|_{w_o^*=0} > \frac{p_c^* p_o'^* - p_o^* p_c'^*}{p_c'^* - p_o'^*}.$$

In the above condition, note that the numerator is negative and the denominator is positive. Thus the choice of  $w_c^* > 0$  by the firm is optimal even when  $\left. \frac{dEV}{dw_c} \right|_{w_o^*=0} < 0$ . This shows the  $IR$  constraint forces the firm to choose bonuses that would otherwise be suboptimal.

Using Equation 14,  $\left. \frac{dEV}{dw_c} \right|_{w_o^*=0}$  can be rewritten as  $(MR^* - p_c' w_c^*) \frac{dx^*}{dw_c} - p_c^*$ . Further, substituting  $w_c^* = \frac{u-Q^*}{p_c^*}$  and using Lemma 2 to substitute for  $\left. \frac{dx^*}{dw_c} \right|_{w_o^*=0}$ , we find that  $w_c^* = \frac{u-Q^*}{p_c^*}$  and  $w_o^* = 0$  is a consistent solution iff  $MR^* < -\frac{(1-\frac{p_c^*}{p_o^*})A^*}{1-\frac{p_c^*}{p_o^*}} - \frac{u-Q^*}{p_o^*} \left( p_o'^* - \frac{(p_o^* - p_c^*) p_o''^*}{p_o'^* - p_c'^*} \right)$ .

The proof for Case (II) is similar.

The proof for Case (III) also proceeds along similar lines, and the requirement for both  $w_c^* > 0$  and  $w_o^* > 0$  is that  $\left. \frac{dEV}{dw_o} \right|_{w_c^*, w_o^*} = \left. \frac{dEV}{dw_c} \right|_{w_c^*, w_o^*}$ . Using this we get the condition specified in Case (III).

While we cannot prove that the solutions specified by all these three cases must exist due to the generality of our specifications and the algebraic complexity of the expressions, the illustrations in the main body of the paper show that these may indeed exist.

LEMMA 5. *The effort choice in equilibrium when the IR constraint is not binding and  $w_c^* > 0$ ,  $w_o^* = 0$  is lower than the socially optimal effort division.*

Since  $w_o^* = 0$ , the firm picks  $w_c^*$  to maximize  $EV$ , i.e.  $w_c^*$  is obtained by solving  $\frac{dEV}{dw_c} = 0$ . Now  $\frac{dEV}{dw_c} = EV'_t(x^*) \frac{dx^*}{dw_c} - p_c^*$ , where  $x^*$  is the incentive compatible effort choice of the programmer. From Lemma 2, we know that  $\frac{dx^*}{dw_c} > 0$ . Therefore, we must have  $EV'_t(x^*) > 0$  at the equilibrium effort choice.

LEMMA 6. *The effort choice in equilibrium when the IR constraint is not binding and  $w_o^* > 0$ ,  $w_c^* = 0$  is higher than the socially optimal effort division.*

The proof of Lemma 6 is similar to that of Lemma 5 and is omitted here.