

HUMAN CAPITAL DEVELOPMENT FOR PROGRAMMERS USING OPEN SOURCE SOFTWARE¹

Amit Mehra

Indian School of Business, Hyderabad, INDIA {Amit_Mehra@isb.edu}

Vijay Mookerjee

School of Management, University of Texas at Dallas, Richardson, TX 75080 U.S.A. {vijaym@utdallas.edu}

A firm can upgrade relevant skills of its programmers by ensuring their participation in carefully chosen open source projects. Highly skilled programmers are more valuable for the firm but participating in open source projects reduces the time they spend doing the firm's projects. This tradeoff determines the optimal extent of programmer participation in open source for the firm. The extent of open source participation may also be influenced by the minimum compensation that must be paid to hire a programmer in the labor market. This is because providing better skills is a way of compensating the programmers by improving their future market value. Hence the firm may want to increase open source participation to keep direct wage payments in check. We develop an analytical model based on optimal control theory to characterize the employment contract that features the best mix of open source participation and wage payments. We also find that the firm benefits more from the presence of open source in a tight labor market (i.e., when programmers have good options besides the employment offered by the firm). On the other hand, programmers are compensated better in the presence of open source opportunities when they have few outside options. This benefit is more for less skilled programmers.

Keywords: Human capital, open source software, employment contracts, training, skill development incentives

Introduction

Need for Human Capital Development in Firms

Today's business environment is marked by intensification of competition, rapid advances in technology, and quick shifts in customer preferences. This requires organizations to be adaptive and flexible. Companies that develop these characteristics are able to adapt to the unpredictable more quickly and

effectively than their competitors. It is well known that these goals can be achieved only if the company becomes a learning organization (Garvin et al. 2008). Human capital literature recognizes that learning can be either formal or informal. Formal training is structured, often classroom based with an instructor planning, implementing, and evaluating the learning taking place (Merriam and Caffarella 1991). Informal training, on the other hand, is not structured and happens through learning opportunities in the workplace environment (Cable and Parsons 2001). It includes learning through mentoring, learning through mistakes and by trial and error, etc., and occurs when the learner is faced with an event or situation that is challenging and nonroutine (Arrow 1962; Becker 1962; Blaug 1976; Larson 1991). The United States government estimates that 70 percent or more of work related training happens through informal training, yet organizations over-

¹Ravi Bapna was the accepting senior editor for this paper. Ming Fan served as the associate editor.

The appendix for this paper is located in the "Online Supplements" section of the *MIS Quarterly's* website (<http://www.misq.org>).

whelmingly invest in formal training. The reason could be that there is no department in charge of informal training initiatives, and further no budget is typically earmarked for informal training (Baldwin-Evans 2006). Hence there is an opportunity for organizations to leverage informal training more effectively in order to achieve the objective of becoming learning organizations (Baldwin-Evans 2007; Conlon 2004; Center for Workforce Development 1998). Thus the problem of creating informal training mechanisms within organizations is contemporary and important, and even more significant for organizations that cannot afford the high cost of formal training programs like in-class training (Conlon 2004).

Open Source Software as an Informal Learning Mechanism

The value of informal learning or learning by doing open source has been empirically confirmed by Hann et al. 2006 in their study of the Apache project. This learning takes place due to an active peer review process (Moody 2002; Raymond 2001; Wayner 2000). Such reviews evaluate code submissions, uncover defects in programming, and suggest ways to make the code more efficient (von Krogh et al. 2003). In addition, programmers who do peer reviews find out about the problems that others have faced and this knowledge helps them manage their own code better (Lakhani and von Hippel 2003). Thus peer review interactions benefit both programmers who help others and programmers who get help from others.

Hertel et al. (2003), Lakhani and von Hippel (2003), and Lakhani and Wolf (2005) find that learning by doing open source projects provides unique learning opportunities for programmers. While formal training programs like classroom training may be a good way to learn *basic skills* like a new programming language, learning by doing may be a more effective means to learn *advanced applications*. An indication that advanced skills are better learned in open source projects is documented in the final report of a 2006 study sponsored by the European Union and conducted by the United Nations University UNU-MERIT in The Netherlands. Further, the skill learning by doing open source projects may be richer than the skill learning by doing closed source projects within the firm. The reason is that the nature of the job content significantly determines the extent of learning (Berg and Chyung 2008). Garavan (1987) suggests that this may be a limitation since the job content may not provide sufficient challenge, variation, and opportunities for learning. Meizsner et al. (2008) emphasize that much of the skill development by doing open source work happens because this work provides challenging novel tasks to programmers.

Hence, firms can overcome the limitation of learning from the current job by enriching the job content of programmers by allowing them to participate in open source project communities. This is in line with the suggestion that informal training at the workplace can be achieved through employee participation in communities of practice (Harvard Management Update 2000). The opportunity to work across firm boundaries creates a unique informal learning mechanism. In this mechanism, knowledge sharing is constrained neither by the numbers and knowledge profiles of employees within the firm nor by the incentives for knowledge hoarding to maintain one's importance within the firm (Lee and Cole 2003). Finally, open source work can be used to easily signal one's skill level to future employers because the code is open for inspection. Consequently, adoption of an informal training program through open source work is likely to be accepted easily by employees without any resistance (Bishop 1991). Thus, learning by doing open source projects constitutes a significant informal training mechanism.

There is some evidence that firms have begun to leverage open source as a skill building device for its programmers. Thus the UNU-MERIT study finds that several companies in retailing, automotive, tourism, and other areas allow their programming staff to do open source work *during work hours* even though their core business is not in a position to get any direct commercial benefits out of such work. The only reason for such behavior is that their programmers learn useful skills that make them more productive on the firm's projects. Schilling et al. (2003) and Boh et al. (2007) also support this observation, finding that diverse experience in related systems improves programmers' learning and thereby increases productivity. For example, programmers who work on commercial game software are likely to gain significant learning advantages by working on open source game software.² The learning from open source may also be important for the firms since the external knowledge it provides may prove valuable (Menon and Pfeffer 2003).³

Open source participation by programmers for learning skills requires them to multitask across open and closed source projects. This needs a holistic organizational set-up (signified by multitasking of work) *vis-à-vis* the Tayloristic set-up (signified by specialization of work). It has been empirically established that the holistic form of organization is more

²A list of open source games is available at http://en.wikipedia.org/wiki/List_of_open_source_games.

³Tim O'Reilly, an expert in the open source area, recommended to Microsoft that it encourage its programmers to participate in open source for exactly this reason (http://oreilly.com/news/osconint_0601.html).

productive (Cartensen 2002) and the adoption of the holistic organizational form has also been documented (Boucekkine and Crifo 2008). This move to the holistic organizational form could be driven by several factors such as the widening of human capital (ability to acquire a variety of skills), presence of inter-task complementarities, better information and communication systems, and workers' preferences for versatile job profiles (Lindbeck and Snower 2000; Pautrel 2004) as well as the firm's products and industry (Gibbs et al. 2010). Typically, the literature in multitasking did not focus on inter-task learning issues and analyzed other questions like impact on effort allocation between tasks due to the nature of incentive contracts (Holmstrom and Milgrom 1991) and tradeoffs between returns to specialization and cost of coordinating the activities of different workers (Bolton and Dewatripont 1994; Becker and Murphy 1992; Yang and Borland 1991). However, Lindbeck and Snower (1996, 2001) did study the impact of inter-task learning and concluded that moving to the holistic organizational form will result in nonuniformity of wages for the same tasks. Further, the wage gap between holistic and tayloristic organizations will widen. Ryu et al. (2005) and Killingsworth (1982) also consider the multitasking situation with inter-task learning and analyze how workers will split their effort across multiple tasks to maximize their own benefit from learning.

Operationalizing Open Source Learning Within Firms

The extant literature does not consider the operational aspect of allocating effort across multiple tasks in order to maximize the firm's benefit from employing the worker. Our paper addresses this gap in the particular context where multitasking arises due to the allocation of programmers' effort between a firm's commercial closed source project and an open source project in order to leverage the learning by doing effects from open source work for enhancing skills and hence productive capacity. Such a firm must also decide whether it should allow its programmers to focus exclusively on open source work for some duration, or whether it should allow them to simultaneously work on internal and open source projects (i.e., *sequential* versus *concurrent* open and closed source work). This question is pertinent in the context of the software industry where firms (e.g., Infosys) often set aside some time for formal full-time classroom training; for example, when a programmer first joins the firm. The question then is whether it is a good idea to let programmers work on open source software in a pattern similar to formal company training programs, or if there is a better alternative. Our analysis is comprehensive since we thoroughly consider all possible scenarios of the efficacy of open source software for

learning and the impact of labor market pressures in hiring programmers. The only other paper that addresses similar issues is Mehra et al. (2011) where the major difference from our paper is that the firms are unable to monitor their programmers.

In the next section of this paper, we analyze a general model where a firm decides on an open source participation-based training scheme for its programmers to maximize their productivity for the firm. We then consider a more specialized model that, while retaining the essential features of skill building, permits us to do a detailed analysis of programmer training through open source when programmer compensation must be increased due to labor market pressures. We discuss the impact of open source on the net value derived by the firm from the employment contract and on the compensation received by programmers. Here we also provide additional insights into how the contract between the firm and the programmer ought to be managed. Finally, we provide conclusions based on the analysis and list limitations and possible extensions. For the sake of brevity, all proofs are relegated to the Appendix.

A General Model and Analysis

In this section, we describe the details of a general model of how open source participation can be leveraged by a firm for training purposes. At this stage, we examine the problem without making specific assumptions concerning the various functional forms that govern the skill building phenomenon. The analysis in this section is, therefore, of a structural nature, where we are able to identify the optimal form of allocating the programmer's time between the firm's closed source project and an open source project that is chosen by the firm to provide training opportunities to its programmers. In the following section, we propose a specialized form of the general model that affords a more detailed analysis.

Model Details

A firm hires a programmer at the beginning of a planning horizon, T , representing the duration of affiliation between the firm and the programmer, or the estimated length of time after which a programmer graduates from programming roles to managerial roles. The hired programmer never leaves the firm before this duration is over. While we do not explicitly enforce this in our model, it is assumed that there exist sufficient contract enforcing mechanisms (such as financial options, or a vesting period) that impose financial losses on the programmer if she leaves prematurely.

The programmer is employed to work on the firm's projects and, additionally, released to work on external, open source projects that are related to the programmer's domain of work. The programmer has an initial skill level $x_0 \geq 0$ that increases during the contract period from working on internal and open source projects and decreases from natural skill depreciation or from the obsolescence of existing skills. We represent the instantaneous skill level by $x(t)$ and the allocation of effort to open source at time t by $0 \leq v(t) \leq 1$. Consequently, the allocation of effort to closed source work at time t is $1 - v(t)$. The skill learning rate from a project is proportional to the instantaneous effort exerted on that project. Accordingly, we model the skill learning rate from closed source work to be $L_c(x(t),t)(1 - v(t))$ and from open source work to be $L_o(x(t),t)v(t)$, where $L_c(x(t),t)$ and $L_o(x(t),t)$ represent the instantaneous skill learning potential from closed source and open source work, respectively. Taking the rate of skill attrition to be $\beta(x(t),t)$, the rate of change of skills ($\dot{x}(t)$) can be represented by the differential equation $\dot{x}(t) = L_c(x(t),t)(1 - v(t)) + L_o(x(t),t)v(t) - \beta(x(t),t)$. This can be simplified to

$$\dot{x}(t) = \delta(x(t),t) + \alpha(x(t),t)v(t) - \beta(x(t),t) \tag{1}$$

where $\delta(x(t),t) = L_c(x(t),t)$ and $\alpha(x(t),t) = L_o(x(t),t) - L_c(x(t),t)$. The above is a fairly general model of the dynamics of skill learning.

The term $\delta(x(t),t) > 0$ captures the minimum skill acquisition rate; a rate of increase in skill when the programmer works on only internal, closed source projects, corresponding to $v(t) = 0$ (i.e., there is no release for open source work).

The term $\alpha(x(t),t)v(t)$ incorporates the increase in the skill acquisition rate from doing open source projects with an allocation fraction of $v(t)$ at time t . The rate of learning depends upon the current skill level of the programmer, the time at which the learning activity is taking place, and the allocation of effort to open source projects. A high level of skill may be conducive to learning in situations when new skills can be picked up more easily because of existing skills. On the other hand, it is possible that a high level of skill may mean that the skills left to learn are also the more difficult ones and hence the skill acquisition rate becomes lower. Thus $\alpha_x(x(t),t)$ may vary with $x(t)$ and may be positive or negative. Similar to the impact of skills, the learning rate may be different at different times, for example, learning opportunities may change with the phase of the project. Hence $\alpha_t(x(t),t)$ can also change with time and be positive or negative. Finally, we require that

$\alpha(x(t),t) > 0$, implying that the rate of skill building increases with the effort exerted in open source projects.⁴

The third term in the equation, $\beta(x(t),t)$, incorporates the rate of attrition of skills. Hence, $\beta(x(t),t) > 0$. Typically, one may expect the skill attrition rate to increase with the skill level since it is often more difficult to stay current at the cutting-edge of an information technology area. Hence a high skill level can get eroded quickly. However, more basic skills within a field change relatively slowly. Therefore, the depreciation rate at lower levels of skill is slower. Such a model of skill attrition is quite common in the human capital literature (Killingsworth 1982). At this stage of the analysis, however, we don't specify the exact functional form of $\beta(x(t),t)$. Note that the skill attrition rate could be time dependent and that the equation for rate of change of skills is programmer specific (i.e., the functions $\delta(x(t),t)$, $\alpha(x(t),t)$, and $\beta(x(t),t)$ can incorporate programmer characteristics). Hence our model is more general than earlier work in human capital literature (e.g., Killingsworth 1982; Sheshinski 1968).

In order to hire a programmer, the firm must satisfy an individual rationality (IR) constraint that reflects the value of outside options for the programmer. We assume that a programmer with an initial skill level of x_0 has an outside option with a market value of M_0 . This means that if the programmer does not accept the firm's contract and takes other opportunities offered in the labor market, she expects to have lifetime earnings of M_0 . Hence employment with the firm must result in lifetime earnings of at least as much as M_0 . x_0 and M_0 can be freely chosen but in most real world situations one can expect that the value of M_0 is increasing in x_0 . We assume that the firm pays the market wage rate to the programmers, which is normalized to zero.⁵ To increase the value of the firm's contract for the programmers, the firm may either pay a wage premium $w(t)$ and/or leverage the programmer's future valuation of skills at the end of the employment contract with the firm. The IR constraint for the programmer that represents the impact of the labor market wage pressure for the firm can, therefore, be formally written as

$$\int_0^T w(t)dt + S[x(T)] \geq M_0 \tag{2}$$

⁴Note that if the learning opportunities from open source are inferior to those obtained from the firm's internal projects (i.e., $\alpha(x(t),t) \leq 0$), the firm would get no advantage from releasing its programmers to work on open source projects, and $v(t)$ will be zero $\forall t$ in the optimal solution.

⁵Alternatively, we could assume a value of $M_0 + w_m$ as the market value of the outside option and, for realism, require that the wage rate paid by the firm must equal or exceed w_m . For convenience, the value of w_m is normalized to zero in the analysis.

where $S[x(T)]$ represents the programmer's *salvage value* of skills. We assume $S[x(T)] = ax(T)$, with $a > 0$, reflecting the fact that higher skills at the end of the contract translate to more value for the programmer since she can get higher paying employment at the end of the contract on the strength of a better skill base.

The instantaneous value of output produced by a programmer for the firm is a function of her skills and effort. Let K represent the marginal value of the programmers output to the firm. Then the production function of the programmer's output for the firm is $K(1 - v(t))x(t)$, where $1 - v(t)$ represents the instantaneous fraction of the programmer's capacity that is allocated to work on internal projects. Consequently the programmer expends the fraction (of productive capacity) $v(t)$ at time t for doing open source work. This production function reflects that the effort of a highly skilled programmer provides more value than that of a less skilled programmer. It also incorporates the fact that if the programmer is allowed more release to do open source projects (larger $v(t)$), it reduces the internal value generated since there is less capacity left to work on the firm's projects. For brevity, we drop reference to the variables as functions of t henceforth. However, this representation sometimes reappears depending upon the context.

The firm's decision problem can now be stated as

$$\begin{aligned} & \max_{w,v} \int_0^T (K(1-v)x - w)dt \\ \text{s.t. } & \dot{x}(t) = \delta(x,t) + \alpha(x,t)v(t) - \beta(x,t) \\ & \int_0^T wdt + ax(T) \geq M_0 \\ & x(0) = x_0 \end{aligned}$$

Here the function being maximized represents the net value (value created minus wages paid) created for the firm by hiring the programmer. The constraints on maximizing this function come from the limitations on skill building rate, the need to satisfy the individual rationality of the programmer, and her initial skill level.

Analysis

The optimization for the firm is a control problem and we use Pontryagin's point-wise maximization method to solve it (Sethi and Thompson 2000). The inequality condition can be rewritten as $\int_0^T [w + \alpha\dot{x}]dt \geq m_0$, where $m_0 = M_0 - ax_0$. The left side of this inequality can be replaced with a new state variable y such that $y = \int_0^t [w + \alpha\dot{x}]dt$. Hence we can write the state equation for the variable y as $\dot{y} = w + \alpha\dot{x}$ with the end

conditions as $y(0) = 0$ and $y(T) = \int_0^T [w + \alpha\dot{x}]dt$. The modified point-wise maximization problem, therefore, is

$$H = K(1 - v)x - w + \lambda(\delta + \alpha v - \beta) + \mu(w + a(\delta + \alpha v - \beta))$$

$$\begin{aligned} \text{s.t. } & \lambda \geq 0 \\ & \mu \geq 0 \\ & \mu(T)(y(T) - m_0) = 0 \text{ (constraint qualification condition)} \\ & y(0) = 0 \\ & y(T) \geq m_0 \\ & x(0) = x_0 \end{aligned}$$

where H represents the Hamiltonian. Note that similar to the state variables, x and y , the co-state variables, λ and μ (for a detailed discussion of the co-state variables, please refer to Sethi and Thompson 2000), are also functions of time. The first order conditions of the Hamiltonian with respect to the control variables w and v are

$$\frac{\partial H}{\partial w} = -1 + \mu \tag{3}$$

$$\frac{\partial H}{\partial v} = -Kx + \lambda\alpha + \mu\alpha\alpha \tag{4}$$

Both first-order derivatives are independent of their respective controls. Consequently if the sign of a derivative is negative, the optimal solution for the corresponding control must be the minimum possible value of that control. On the other hand, if the sign is positive, the optimal control must take the maximum possible value. Thus we have a bang-bang (corner) solution for the control. In case a first-order derivative is zero, the optimal value of the corresponding control can be different from its maximum and minimum values (i.e., the control may take on a value between its maximum and the minimum). Such values of control are termed singular.

Note that if a singular solution exists for the control v , we can conclude that a regime where programmers work concurrently in the firm's project and the open source project is optimal. If such solutions do not exist, then the optimal value of control v must be either its maximum value of 1, or its minimum value of 0. Then we can conclude that a regime where programmers work exclusively in either the open or the closed projects at any instant is optimal. Next, we turn to characterizing the optimal solutions of the control variables w and v throughout the contract duration.

First, we focus on w . The following result helps characterize the control w in the whole contract duration.

LEMMA 1. *Either $w = \text{constant} > 0 \forall t \in [0, T]$, or $w = 0 \forall t \in [0, T]$*

Lemma 1 shows that the firm pays either a constant or a zero wage premium in the entire contract duration. Thus the wage premium rate solution is very simple for the firm to implement. This is not surprising since it is only the total amount of the wage premium paid over the contract duration that matters in order to satisfy the individual rationality (IR) constraint. If such a wage premium is to be paid, the firm can pay a wage premium at a constant rate to meet the overall requirement. Otherwise, there is no need to pay any wage premium at all. We will later discuss the extent of the wage premium payment that the firm must pay to satisfy the programmer's IR constraint.

Next, we turn our attention to the behavior of the control throughout the contract duration. The next result is useful in this regard.

LEMMA 2. *The value of $\frac{\partial H}{\partial v}$ is equal to zero in at most one contiguous region during the contract period, T .*

Note that Lemma 2 cuts down the solution structures dramatically. For example, no solution where the control v fluctuates between 0 and 1 for a multiple number of times can be optimal. The only situation in which the value of $\frac{\partial H}{\partial v}$ becomes zero is one where a singular solution exists. Hence v can be between 0 and 1 in just this interval. Since $\frac{\partial H}{\partial v}$ is a continuous function in t , it can be either positive or negative prior to being 0. Correspondingly, $v = 1$, or $v = 0$, in this region. For similar reasons, after the singular region, $v = 1$ or $v = 0$.

The insights provided by Lemma 2 allow us to further characterize the solution in terms of an understanding of the circumstances under which a particular value for v is chosen. Further, we must also obtain the value of the control v in the singular solution. These issues are resolved in the following proposition.

PROPOSITION 1. *The appropriate choice of effort split between open and closed source work during the contract duration (T) can be divided into three phases as follows:*

- (a) **(Initial phase)** *At the contract commencement ($t < \bar{t}$), the firm chooses either full effort in closed source or full effort in open source ($v = 0$ or $v = 1$).*
- (b) **(Intermediate phase)** *In the intermediate stage of the contract ($\bar{t} < t < \hat{t}$), the firm chooses a split of programmer effort, $0 < v < 1$, between open and closed source work.*
- (c) **(End phase)** *Toward the end of the contract ($\hat{t} < t < T$), the firm chooses full effort in closed source if the IR*

constraint does not bind, and full effort in either closed source or open source ($v = 0$, or $v = 1$) if the IR constraint binds.

Proposition 1 combines the insights provided by lemmas 1 and 2. The major outcome of this proposition is that the contract period may be subdivided into a maximum of three phases. Thus, depending on the parameter values, it is possible that one or even two of these phases may be missing in the optimal solution. The initial phase is marked by full effort in either closed source or open source work. The intermediate phase is characterized by a split of effort between open and closed source. Finally, the end phase is again marked by full effort in either closed source or open source work. The choice of the split v in the intermediate phase essentially reflects the firm's choice between the future and current productivity of the programmer. A larger fraction of time spent in doing open source work makes the programmer more skillful in the future at the cost of producing less value in the present. By asking the programmer to do only open source work or only closed source work, the firm focuses entirely on either its future productivity or its current productivity. This is generally not the best thing to do since the firm can do better by aligning the importance of both current and future productivity. An exception to this rule is likely to arise at the beginning of the contract because the firm may want an appropriate level of skills before it starts balancing between current and future productivity. Another exception may arise toward the end of the contract. In this situation, the future productivity of skills becomes less important because of the limited period left before contract expiry. Consequently, the firm may again do better by focusing on maximizing the current productivity only ($v = 0$). However, when the market gives a high value to programmers, the firm may choose to compensate the programmer by providing her with a more valued skill portfolio at the end of its contract with the programmer. This can be done by using the period near the end of the contract to accommodate skill creation for the programmer by completely releasing her for open source work ($v = 1$). In general, the firm can meet the IR constraint of the programmer when it binds in one of the four ways enumerated below.

1. Increase wages only.
2. Increase training only. This can be done by increasing the duration for which the programmer is fully released to do open source work (possibly in the time intervals $t < \bar{t}$ and $t > \hat{t}$). Alternatively, the concurrent training region (in the time interval $\bar{t} < t < \hat{t}$) where both open source and closed source programming are simultaneously done can also be extended.

3. Increase both training and wages. In this case part of the constraint is met by increasing the training, which results in higher programmer skills at the conclusion of the contract. The rest of it is met by increasing the wages.
4. Decrease training and increase wages. If this is the solution, the firm trains the programmer less as compared to the case when the IR constraint does not bind. All the shortfall in satisfying the IR constraint is met through wages.

In general, the choice between utilizing training and wages to meet the programmer’s rationality constraint depends upon the tradeoff between productivity loss by increasing training (since less time is available to produce code for the firm’s commercial project) and the direct cost of wages incurred by reducing training (reducing the training causes the programmer to end the contract with lower skills, hence additional wages must be paid to meet the shortfall).

While some details of managing programmer compensation through wages and training through open source release time are provided in the analysis in this section, several important details (e.g., how to manipulate the training duration or how to change the optimal mix of training and wages to satisfy the programmer’s IR constraint) could not be determined using the general form of the model analyzed in this section. Hence we study these effects under a special functional form of δ , α , and β . This analysis follows in the next section.

Analysis of a Specialized Model

In this section, we present results obtained from the analysis of a specialized model. Here we are able to provide expressions for the start and end of the intermediate phase of the contract as well as clarify the impact on the optimal training and wage premium payment policies when the value of outside options for the programmer are sufficiently attractive (IR constraint binding). We specialize the learning from closed source as $L_c(x,t) = z_1 + z_2x$, from open source as $L_o(x,t) = z_3 + z_4x$, and the skill attrition rate as $\beta(x,t) = z_5x$, where $z_1, z_2, z_3, z_4, z_5 > 0$. While this formulation is more specialized than the general form used earlier, it nevertheless incorporates important features such as skills acquisition rate being dependent upon the current skill level of the programmers. Of course, our analysis can also be done by not incorporating the dependence of learning on skills (i.e., setting z_2 and z_4 equal to zero). The skill building equation (Equation 1) can now be rewritten as $\dot{x}(t) = z_1 + (z_3 - z_1)v + (z_4 - z_2)xv - (z_5 - z_2)x$. Writing $z_1 = c_1, z_3 - z_1 = c_2, z_4 - z_2 = c_2$, and $z_5 - z_2 = c_4$, we can write Equation 1 in a further simplified form as

$$\dot{x} = c_1 + c_2v + c_3vx - c_4x \tag{5}$$

If the differential in skill learning rate is in favor of the open source project, the firm may choose to utilize part of its programmer’s time in doing these projects. In order to enforce superior learning rate from open source projects, we assume $c_1 > 0$ and $c_3 > 0$. Further, note that the skill deterioration rate is not considered to be merely a constant but increasing with the current skill level ($z_5 x$). If this is not so, the skill acquisition rate may become positive once the current skill level x is sufficiently high. This leads to the unrealistic situation that the skills can potentially reach a level of infinity. Previous work in human capital development also considers skill depreciation in a similar way (Killingsworth 1982). Note that we also need $c_4 > c_2$ to ensure that the skill level is bounded. If this is not so, the skills will potentially reach a level of infinity, which is again not realistic. Thus our simplified formulation captures the essential spirit of the problem in that there is a differential in the skill building rate with learning on the open and closed source projects and that there is skill attrition that increases with the current skill level of the programmer.

To facilitate the analysis we restrict the parameter space to $K > \frac{ac_3(c_1c_2 + c_2c_4)}{c_1c_3 + c_2c_4 - \sqrt{c_2(c_4 - c_3)}(c_1c_3 + c_2c_4)}$ and also $c_1 < \frac{c_2c_4}{c_4 - c_3}$ which can be restated as $z_1 < \frac{z_3z_5}{2z_5 - z_4}$. The earlier condition implies

that the firm has a sufficiently high marginal value of programmer productivity (i.e., the programmer produces sufficient value while doing internal projects). While the problem can be solved without imposing this restriction, it limits the number of different cases that need to be considered and focuses attention on the main issues without getting lost in details. The later condition compares the learning from closed and open source and says that the value of learning from open source is bigger than a particular threshold which is defined by the learning rate from closed source. We relax this condition in the subsection below. The specialized model can now be analyzed to provide additional results that characterize the variable v , thereby refining the insights presented in Proposition 1.

LEMMA 3. *If $\frac{\partial H}{\partial v} = 0$ for some $t \in [\bar{t}, \hat{t}]$ where $0 < \bar{t} \leq \hat{t} < T$, then we have*

1. (Initial phase) For $0 < t \leq \bar{t}$,
 - a. $x_0 < \bar{x} \Rightarrow v = 1$.
 - b. $x_0 > \bar{x} \Rightarrow v = 0$.
2. (Intermediate phase) For $\bar{t} < t \leq \hat{t}, v = \bar{v}$.

3. (End phase) For $\hat{t} < t \leq T, v = 0$.

$$w \text{ h e r e } \quad \bar{x} = \frac{\sqrt{c_2(c_4 - c_3)(c_1c_3 + c_2c_4)} - c_2(c_4 - c_3)}{c_3(c_4 - c_3)} \quad \text{a n d}$$

$$\bar{v} = \frac{c_2c_4 - \sqrt{c_2(c_4 - c_3)(c_1c_3 + c_2c_4)}}{c_2c_3}$$

Note that Lemma 3 enables us to express the choice of v throughout the contract duration T as a function of the initial skills. It refines the result in Proposition 1 because we now see that in the final phase of the contract, full effort will only be dedicated to closed source and never to open source projects, even when the IR constraint is binding. This improves our understanding of how to split the programmer’s effort optimally between open and closed source work. Note that it is easy to implement the insights obtained here since the optimal choice of the effort v depends only upon a threshold value of a single criterion, the initial skill level of a programmer. The firm is likely to have a good estimate of the skill level of an incoming programmer since the recruiting process is mainly geared toward evaluating this parameter. Further, the optimal policy does not need to vary continuously with the initial skill level; it is only necessary to estimate whether the incoming skill level is lower or higher than the skill level \bar{x} .

Next we focus on characterizing the cutoff times instants, \bar{t} and \hat{t} . While the algebraic expressions for both the cutoffs are obtainable, these expressions are tedious (see Appendix). Here, we present results that capture the impact of the individual rationality constraint on these cutoffs.

PROPOSITION 2. *When the learning effect of open source is larger than a threshold $\left(\frac{c_2c_4}{c_4 - c_3} > c_1\right)$, the beginning of the intermediate region of the contract (with effort split between open and closed source work) is unaffected by wage pressure from the labor market of programmers. However, the termination of the intermediate region of the contract is extended due to this wage pressure.*

The beginning of the intermediate contract region is unaffected by the wage pressures from the market since this point is dictated completely by the firm’s considerations of maximizing the current and future productivity. In this region the firm maintains a constant split of effort between open and closed source work and, consequently, a steady skill level of the programmers. However, the wage pressures from the market become important near the end of the contract since the firm may meet the wage demands by altering the effort split in a way that results in higher skill levels for programmers at contract termination. This allows the programmers to

get more highly paid jobs in the future, thereby obviating the need to pay direct wages to the programmers to meet the market wage pressure. Note that the firm normally does not bother about the skills of its programmers toward the end of the contract and dictates full dedication of their time to closed source work ($v = 0$) in order to maximize its productivity. In order to increase programmer skills at contract termination as market pressures go up, the firm may choose to maintain the intermediate region for a longer duration before moving its focus completely to current productivity. In this way, it gives up on some productive output of the programmer toward the end of the contract, but benefits as its programmer has higher skills at the end of the contract.

We now delve into the issue of when exactly the termination of the intermediate region is extended due to market wage pressures. In Proposition 3, we specifically lay out these conditions. Thus we extend the results presented in Lemma 1 to give specific conditions under which w is strictly positive and its value in such cases.

PROPOSITION 3. *When the learning effect of open source is large $\left(\frac{c_2c_4}{c_4 - c_3} > c_1\right)$, the total wage premium paid by the firm is as follows:*

1. *If $M_0 < M_1$, the firm does not pay any wage premium since the market wage pressure is trivially satisfied by the contract termination skills of the programmers.*
2. *If $M_1 < M_0 < M_2$, the firm does not pay any wage premium and meets the increase in market wage pressure by gradually extending the duration of the intermediate phase so that the programmer skills at termination of the contract go up.*
3. *If $M_2 < M_0$, the firm pays a wage premium over and beyond the compensation through programmer skills generated by extending the duration of the intermediate phase to the maximum extent possible (i.e., skills at $M_0 = M_2$).*

The threshold values are $M_1 = \frac{a(c_1c_3 - c_2c_3 + 2c_2c_4 - 2\sqrt{c_2(c_4 - c_3)(c_1c_3 + c_2c_4)}}{c_3^2}$
 and $M_2 = \frac{a((c_1c_3 + c_2(2c_4 - c_3)) - 2\sqrt{c_2(c_4 - c_3)(c_1c_3 + c_2c_4)})K - ac_1c_3^2}{c_3^2(K - ac_4)}$.

Case A in Figure 1 represents the situation where the IR constraint is not binding and Case B shows the situation where the IR constraint binds. As Proposition 2 states, the intermediate phase is extended to increase the programmer skills at contract termination to meet the programmer’s rationality constraint. Thus the firm is able to meet the pro-

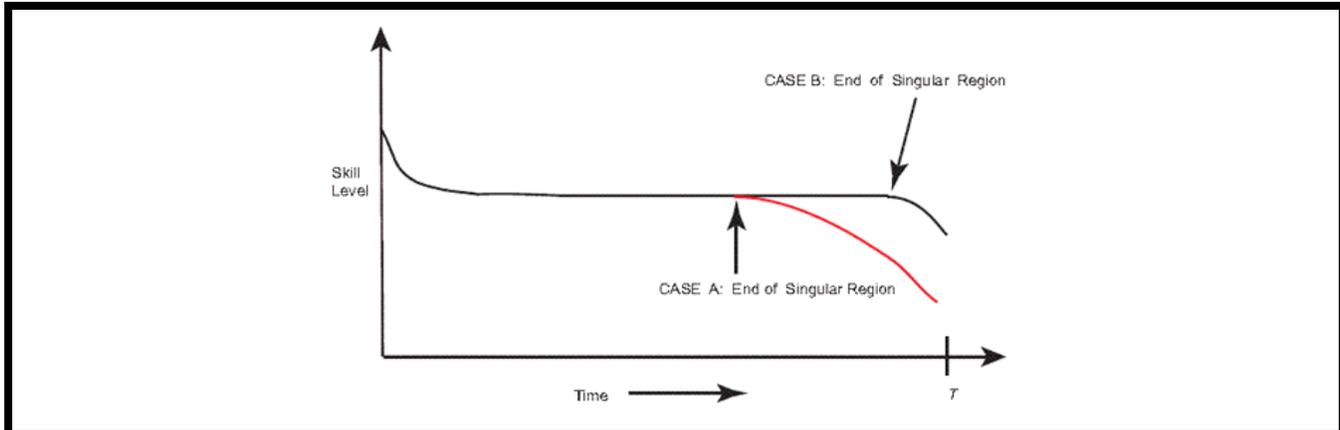


Figure 1. Illustrating Region (b) in Proposition 3

programmer's rationality constraint without paying a wage premium. As the intermediate phase extends, the end phase of the contract is shorter with $v = 0$. In part 3 of Proposition 3, we note that when the value of the outside option increases beyond a point ($M_0 = M_2$) any further extension of the intermediate phase (to impart greater skills at termination of the contract) is no longer cost-efficient and the firm finds it better to pay a positive wage to meet the rationality constraint.

Thus the firm first extends the duration in which concurrent work on open and closed source projects is done to satisfy the rationality constraint and then starts to pay wages when the outside opportunities exceed a threshold value (M_2). This particular sequencing of choices to satisfy the IR constraint of the programmer is a distinguishing feature of our propositions and equips the manager with a fundamental insight on how to compensate programmers as their outside opportunities become more attractive. The reason why this sequencing occurs can be understood from the process in which the firm realizes optimal returns. In the absence of market pressures, the firm optimizes returns from employing a programmer by choosing an intermediate phase with effort split between open and closed source projects. The increase of wage pressure from the market forces the firm to compromise on its productivity by increasing the compensation of the programmer. One simple way to do this is by compensating the programmer directly through a wage premium to the extent needed. But the firm has another option: increasing the duration of the intermediate phase causes an increase of programmer skill level at the conclusion of the contract, thus compensating for the market wage pressure. A small increase in duration of the intermediate phase has minimal impact on the productivity of the firm since the slope of the concave productivity function with respect to duration of the intermediate phase near its

maximum value is nearly zero. As the wage pressure increases, however, the duration of the intermediate phase must be extended more and more to provide greater compensation to the programmers through skills. When a point is reached where the loss in marginal productivity by increasing the duration of the intermediate phase becomes more than the marginal loss due to payment of a direct wage premium, the firm stops extending the intermediate phase and switches to paying a direct wage premium to the programmer to meet market pressure.

The above analysis is based on the particular form of the skill building equation, Equation 5. So, to ascertain that our results are not dependent on this linear form, we conducted a numerical analysis with several other nonlinear forms of the state equation. We found that all of our essential results were preserved, that is, the firm responded to increase of market pressure (IR constraint binding) by extending the intermediate region to increase training through open source resulting in higher skills for the programmer at termination of the contract.

It is also important to point out that solutions with concurrent work on the firm's project and the open source project (singular solutions) may not always exist. The solution with concurrent training specified in Proposition 2 requires that $\bar{x} > 0$, $0 < \bar{v} < 1$, and $\bar{t} \leq \hat{t}$. $\bar{x} > 0$ requires $(c_1 + c_2)c_3 \geq 0$, which is always true due to the requirements on the parameters. $\bar{v} >$

0 requires $c_3(c_1c_4 - c_1c_3 - c_2c_4) < 0 \Rightarrow \frac{c_2c_4}{c_4 - c_3} > c_1$ since $c_3 > 0$.

Thus, the learning from open source must at least be a minimum amount. $\bar{v} < 1$ requires $(c_1 + c_2)(c_3 - c_3) > 0$, which is trivially true. We also need $\bar{t} \leq \hat{t}$. This condition requires a minimum contract duration, that is,

$$T \geq \frac{1}{c_4} \text{Log} \left[\frac{(c_1 - c_4 x_0)(c_1 c_3 (c_3 - c_4) + c_2 (c_3 - 2c_4)c_4 - 2c_4 \sqrt{c_2 (c_4 - c_3)(c_1 c_3 + c_2 c_4)})}{(c_1 (c_3 - c_4) + c_2 c_4)^2} \right]$$

We refer to this threshold as \bar{T} . The solutions with the concurrent open and closed source project work component as presented in this section will break down if any of these conditions are not satisfied. In the following section, we specify the solutions when learning effects from open source are small.

Training with Small Learning Advantage from Open Source

Here, we investigate the situation where open source is not as effective for training purposes, that is, when $\frac{c_2 c_4}{c_4 - c_3} < c_1$. The next result shows that it is still possible to sometimes use open source to build a programmer’s skills while sometimes it is not.

PROPOSITION 4. *When the learning effect of open source is small $\left(\frac{c_2 c_4}{c_4 - c_3} < c_1\right)$, the firm’s policy of utilizing open source project work for learning is as follows:*

1. *If $x_0 > \bar{x}$, the firm chooses to have its programmer work exclusively on the closed source project ($v = 0$). If IR binds, wage premiums are paid to meet the deficit.*
2. *If $x_0 < \bar{x}$, the firm first has its programmer work exclusively on the open source project ($v = 1$) for a time interval τ , then it switches over to having its programmer work exclusively on the closed source project ($v = 0$). If IR binds due to an increase in market pressure, τ is increased to utilize programmer skills to compensate for wages. If the market wage pressure increases beyond a threshold, direct wage premiums are paid to satisfy the market constraint.*

If the learning effect from the open source project is not large enough, the effort split between open and closed source projects in the intermediate phase to maintain balance between current and future productivity becomes suboptimal. The best training strategy for the firm still depends upon the initial skill level of the programmer. If the skill level is higher than the threshold \bar{x} , the firm does best by abandoning the skill building process of the programmer using open source and should simply utilize her full time in producing code for the closed source project. The presence of market pressure on wages does not have an impact on this policy since building skills by using open source is already inefficient and so the firm does not invest in open source to build new skills to meet

the market wages through providing higher skills at contract termination. In this situation, all shortfall in wages is met through direct payment of wage premium.

If the skill levels are lower than the threshold, the impact of taking time away from productive work is small. Consequently, the firm must exclusively utilize open source work to bring the programmer “up-to-speed” by increasing her skills to the level of \bar{x} . Once this level is achieved, the firm must then utilize the programmer full time into closed source projects. The presence of market pressure on wages has an impact on this policy. Increasing market pressure on wages now induces the firm to build a higher skill level for the programmer upfront by utilizing open source project work exclusively for a longer duration. Then the firm moves to fully utilizing the programmer in doing closed source work.

It is useful to note that the sequencing of managerial choice to first increase the open source training and then the wages remains intact. The difference from the previous section is that since there is no intermediate phase with effort split between open and closed source projects, the increase in open source training is brought about by increasing the duration of the initial phase when programmers are deployed to work full-time on open source projects only.

Situations where the contract duration T is too short (i.e., $T < \bar{T}$) may also be possible. However, those results are similar to the ones presented in this section, so we exclude discussion of such cases.

Operational and Welfare Implications ■

In this section we study the impact of various parameter values on the operational details of the contract and also on the benefit to the firm and the programmer with respect to a benchmark employment contract that operates without open source opportunities. We focus attention on cases where the intermediate phase exists since the impact of various parameters in other cases is similar.

Operational Implications

We are particularly interested in analyzing the impact of learning benefits from open and closed source projects encapsulated by the parameters c_1 , c_2 , and c_3 on \bar{x} (skill level in the intermediate phase), \bar{v} (the extent of time devoted to open source projects in the intermediate phase), \bar{t} (the instant at which the intermediate phase begins), and \hat{t} (the instant at which the intermediate phase ends).

First, we consider the impact of these parameters on \bar{x} . Note that \bar{x} increases with c_1, c_2 , and c_3 . Quite expectedly, the skill level in the intermediate phase improves as the rate of learning from either the firm's project or the open source project increases. Further, \bar{v} decreases with c_1 , but increases with c_2 and c_3 . As learning from the firm's project becomes better (increasing c_1), training through open source in the intermediate phase is reduced. On the other hand, as learning through open source becomes more effective (increasing c_2 and c_3), training by utilizing open source becomes more attractive, leading the firm to set more time for doing open source work in the intermediate phase.

The behavior of \bar{t} with respect to the chosen parameters depends upon whether x_0 is more or less than \bar{x} . In case $x_0 < \bar{x}$, the signs of the partial derivatives of \bar{t} with respect to c_1, c_2 , and c_3 can be positive or negative, depending on the parameter values. The reason is as any of these parameters increase, \bar{x} increases (see the discussion in the previous paragraph), thus increasing the gap between the skills in the intermediate phase and the time of entry. This causes an increase in the time required to attain the skills at the intermediate phase. However, increase in any of these parameters increases the rate of skill acquisition in the initial phase ($\dot{x} = c_1 + c_2 + c_3x - c_4x$ as $v = 1$ in the initial phase, please refer Equation 5). This effect reduces the duration needed to attain the skill level (\bar{x}) at the intermediate phase. Overall, the impact depends upon the relative values of the parameters. In case when $x_0 > \bar{x}$, the impact of c_1 on \bar{t} again can be positive or negative but the partial derivatives of \bar{t} with respect to c_2 and c_3 are clearly negative since \bar{x} increases with these two parameters, thus closing the skill gap between the intermediate phase and entry level skills, while the skill acquisition rate is not affected ($\dot{x} = c_1 - c_4x$ as $v = 0$ in the initial phase, please refer Equation 5).

For \hat{t} , we find that it increases with c_2 when IR constraint binds with a positive wage premium, decreases with c_2 when IR constraint binds with zero wage premium, and again increases with c_2 when the IR constraint is non binding. For an example, see Figure 2 ($a = 1, c_1 = 0.2, c_3 = 0.1, c_4 = 0.15, K = 1, T = 24, M_0 = 2$). As c_2 increases, the productivity of the programmer in the intermediate phase increases ($K(1 - \bar{v})\bar{x}$ increases). Further, since \bar{x} increases with c_2 while the rate of skill loss in the end phase is unaffected ($\dot{x} = c_1 - c_4x$ as $v = 0$), the productivity of the programmers in the end phase improves. \hat{t} is adjusted by the firm depending on the relative increases in the productivity in the intermediate and end phases.

Next, we turn to the impact of c_3 on \hat{t} . For an example, see Figure 3 ($a = 1, c_1 = 0.1, c_2 = 0.2, c_4 = 0.15, K = 2, T = 24, M_0 = 1$). The interesting observation here is that one might

expect (by analogy with the effect of c_2) an increase in \hat{t} with c_3 when IR constraint binds with positive wage premium and when the IR constraint is non binding. However, as Figure 3 shows, the impact is actually the reverse and \hat{t} decreases with c_3 in these two regions. Thus, it is important to be careful about how the parameters may impact the end of the intermediate phase. While both c_2 and c_3 seem to be similar since increase in either of these parameters increases the skill acquisition rate, clearly they still may have a different impact on the decision variables because their impact on programmer productivity in the intermediate phase and end phase of the contract may be different. To round off this discussion, we note that just like c_3 , the impact of c_1 on \hat{t} can also be different from c_2 .

Welfare Implications

Consider a benchmark contract where the firm does not allow any open source work because, for example, there are no open source opportunities in the environment in which the firm operates. In such cases the firm will be unable to find an open source project that is sufficiently synergistic in terms of learning benefits with its own projects. In the benchmark contract, if programmer skills at contract termination are high enough so that the programmer's rationality is satisfied, additional wage payments are not required. Otherwise, the firm must pay additional wages to satisfy the rationality constraint.

We first analyze the welfare impact of open source on programmers. If the IR constraint binds, the programmer is simply compensated to the extent required by the IR constraint in both contracts. The situation where the IR constraint does not bind is the most interesting for both contracts (i.e., the benchmark contract and the contract in which open source work is included). In this situation, the net compensation to the programmer is due to skills at contract termination. Solving Equation 5 with $v = 0$ gives the programmers' skill in the benchmark contract at contract termination. The

corresponding value of skills is $\frac{a(c_1(1 - e^{-c_2T}) + c_4x_0)}{c_4}$. The value of skills at contract termination when open source is used is $\frac{a((c_1c_3 - c_2c_3 + 2c_2c_4 - 2\sqrt{c_2(c_4 - c_3)})(c_1c_3 + c_2c_4))}{c_3^2}$. These expressions

show that there is a significant impact of the initial skill level x_0 on the value of skills in the benchmark contract if the contract duration, T , is small while there is no impact in the contract where open source is used. In Figure 4 ($a = 1, c_1 = 0.2, c_2 = 0.3, c_3 = 0.1, c_4 = 0.15, K = 1, T = 2, M_0 = 0$) we plot the ratio of the programmers' compensation in the open source contract versus the benchmark contract. The figure shows

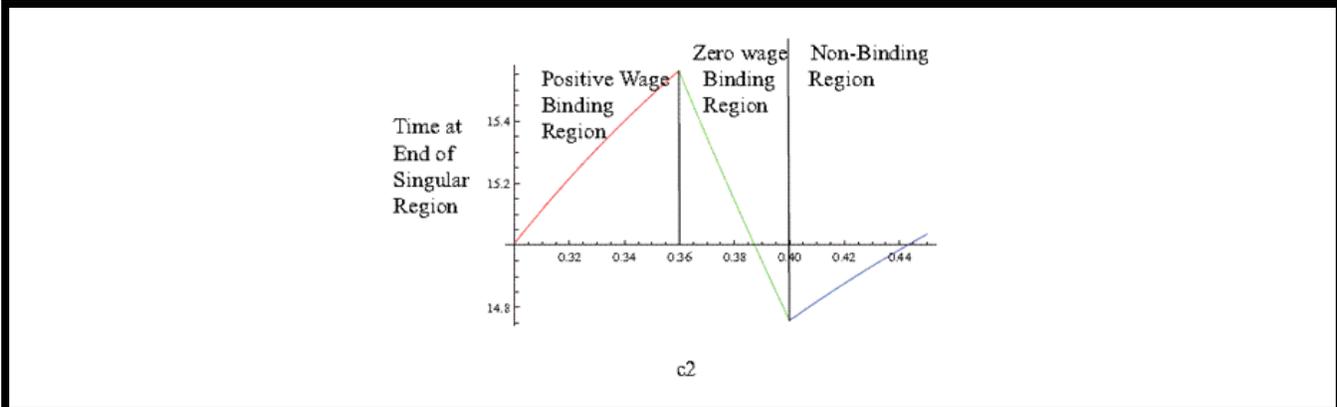


Figure 2. Impact of c_2 on \hat{t}

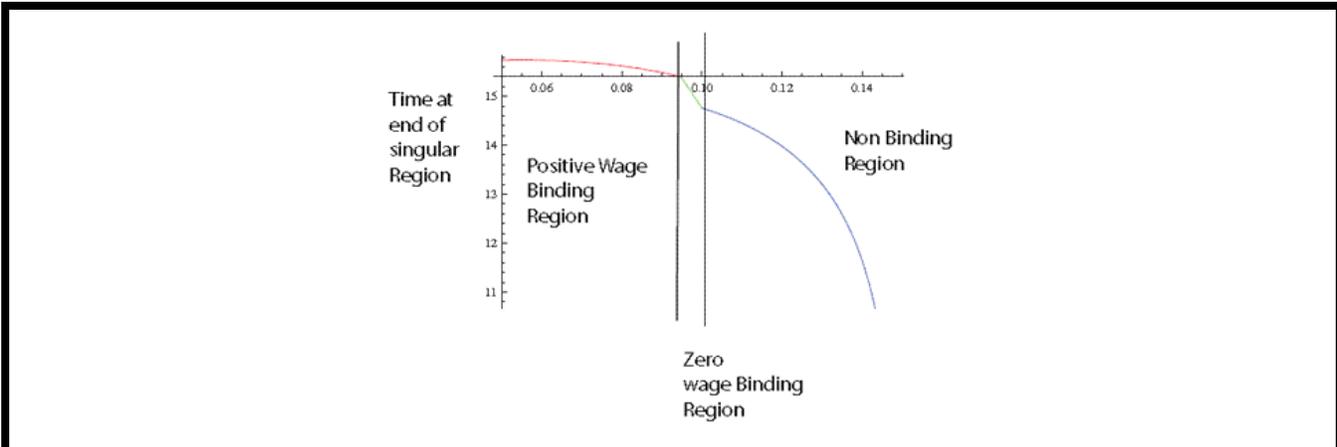


Figure 3. Impact of c_3 on \hat{t}

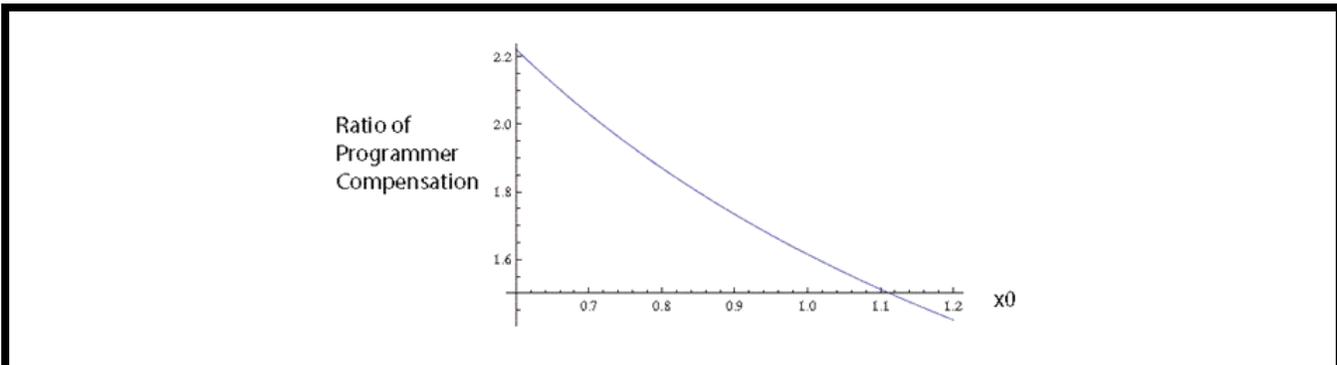


Figure 4. Impact of Initial Skills on Benefitting from Open Source

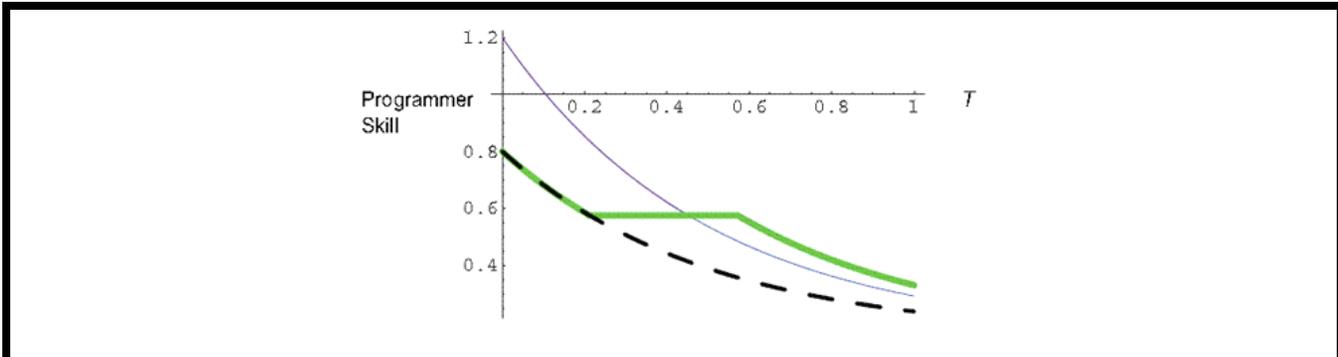


Figure 5. Why Does a Low Skilled Programmer Get a Higher Advantage with Open Source?

that the presence of open source opportunities increases programmer compensation (ratio is greater than 1) and, further, the relative benefits of open source to less skilled programmers are higher.

Why does open source provide more benefits to programmers with low skills? Figure 5 ($a = 1, c_1 = 0.3, c_2 = 2, c_3 = 0, c_4 = 2, K = 1, T = 1$) depicts the case of two programmers with initial skill levels, $x_0 = 0.8$ and $x_0 = 1.2$, with corresponding market values of $M_0 = 0$ and $M_0 = 0.1$, respectively. The skill level for both programmers in the intermediate phase of the contract when open source is allowed is $\bar{x} = 0.6$. Consider the programmer with the initial skill level of 1.2. In accordance with Proposition 1, the firm chooses in the initial phase until the skills degrade to 0.6. Then the skill level is held constant at 0.6 for the duration of the intermediate phase. After the intermediate phase, the firm again chooses $v = 0$ and the skill levels further fall until the end of the contract. The bold line at $T = 1$ depicts the skill at contract termination. In the benchmark contract, the plain line at $T = 1$ depicts the skill at contract termination. The gap between the bold and plain lines at $T = 1$ reflects the advantage of the open source contract for the programmer with initial skills $x_0 = 1.2$. Similarly, the gap between the dashed and the bold lines reflects the advantage of the open source contract to the programmer with initial skills $x_0 = 0.8$. Clearly the gap (advantage) is more for a less skilled programmer. Thus, given a set of other parameter values, the skills at contract termination in the contract that uses open source are always the same irrespective of the initial skills. However, the programmer with low initial skills ends up with lower skills at termination of the contract in the case of the benchmark contract. Consequently, the net gain for a programmer who starts off with low skills is much more if the employment contract incorporates training through open source participation.

Now we analyze the welfare impact of open source learning benefit on firms. This benefit is captured through the parameters c_2 and c_3 . For the firms, the benchmark contract is worth $\int_0^T Kx(t)dt$ while the open source contract is worth $\int_0^i Kx(t)(1-v_1)dt + \int_f^i K\bar{x}(1-\bar{v})dt + \int_f^T Kx(t)dt$, where v_1 is either 0 or 1 depending on whether $x_0 > \bar{x}$ or $x_0 < \bar{x}$. If the programmers' rationality is binding and wages have to be paid, then the worth of a contract reduces by the extent of the necessary wage payment. In Figure 6 we plot the ratio of the worth of the open source contract and the benchmark contract versus c_2 . The parameter values are $a = 1, c_1 = 0.5, c_3 = 0, c_4 = 0.5, K = 1, T = 24, x_0 = 0.1, M_0 = 0$ (non-binding case), $M_0 = 1.3$ (zero wage with IR constraint binding case), and $M_0 = 1.5$ (positive wage with IR constraint binding case). The firm gets the maximum relative benefit in the positive wage with IR binding case, followed by the zero wage IR binding case, followed by the IR non-binding case. The reason is that the only recourse to the firm in the case of the benchmark contract is to pay wages if programmers' rationality binds. In the open source contract, the firm has the additional option of meeting the constraint by improving the skills of the programmer through devoting more time toward open source participation. More choices to meet the constraint lead to better solutions for the firm. In the positive wage IR binding case, the firm actively uses both options, thus improving its benefit. In the zero wage IR binding case, the firm uses only one option, namely, one of providing higher skills at contract termination. Thus the relative benefits are less since the firm uses lower degrees of freedom to compensate the programmer. Finally, in the non-binding case, the constraint is trivial and relative benefits arise only from better learning through open source. The plot of the ratios of the two contracts with respect to c_3 is similar.

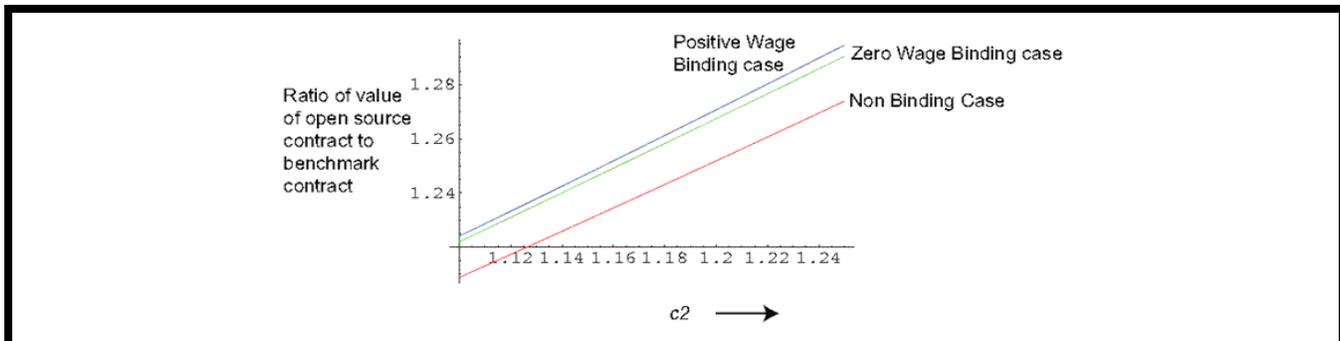


Figure 6. Relative Benefit of Open Source Contract in Binding/Non-binding Regions

Discussion and Conclusions

Results and Managerial Implications

Open source participation can be an effective way to train programmers employed by a firm. The skill gained from participating in open source projects not only improves future employment opportunities for the programmer, but also improves the programmer’s contributions to the internal projects of the firm. By allowing a programmer to participate in open source, a firm foregoes some productive time that would be spent on internal project work today, but gets a more productive programmer to work on its projects in future. Employment contracts must, therefore, be written to balance the current and the future productivity of the programmer. We find that an employment contract that uses open source participation for training programmers typically consists of three phases: an initial phase, an intermediate phase, and an end phase. During the initial phase, the firm either chooses full release or no release for open source projects, depending on the initial skill level of the programmer. A highly skilled programmer is exclusively put to work on internal projects, whereas a relatively less skilled one is allowed full release time for open source participation. During the intermediate phase, only a fraction of the programmer’s time is allowed for open source participation. The end phase invariably concludes with the programmer working exclusively on internal projects.

The labor market constraint also plays an important role in the design of an optimal employment contract. When the labor market wage pressures are high, the firm should seriously consider increasing the exposure of its programmers to open source projects. The firm typically should do this by extending the duration of the intermediate phase. By doing so, the firm provides increased learning to its programmers and boosts their future market value. At the same time, it saves the firm from the cost of paying wages.

Firms derive benefits from the presence of open source due to improved programmer productivity and the extent of this benefit increases with labor market pressures. This is because the increased compensation required due to labor market pressure can be met more efficiently when the firm uses open source since it is an additional compensation mechanism in the form of the release time that can be provided to programmers to participate in open source projects. Programmers with low skills typically enjoy a higher relative benefit in the presence of open source opportunities. This is because the employment contract with open source training is such that the skills at the end of the contract are the same for programmers at all skill levels.

This study has an important message for software firms with specialized products or services. Such firms may tend to restrict programmer participation in open source projects and prefer to pay a wage premium to attract programmers in the face of labor market wage pressures. This practice not only lowers the programmer’s exposure to the learning benefits of open source participation but also increases the firm’s out-of-pocket costs, namely, the wages it must pay to hire the programmer. One reason for this practice may be that it is difficult to find open source projects that are synergistic to the firm’s internal projects and have valuable learning outcomes in the context of the firm’s requirements. Thus both the firm and the programmer would gain if there are more efficient ways to identify open source projects that are a better match for the firm’s internal projects. Currently, however, the documentation and description of open source projects is often poor. This creates inefficiencies for the entire ecosystem because neither do programmers benefit from valuable learning opportunities nor do firms save wage costs or enjoy productivity improvements that could result from open source participation. Thus it appears that there is a win-win opportunity for firms, individual programmers, and the open source community to better disseminate information on open source activity.

Limitations

A limitation of the study is that a specific model of skill-building is used to produce several of the results (see the previous section, "Operational and Welfare Implications"). Without a specific model, it is difficult to arrive at the results presented in the previous section of this paper; however, it is important to recognize that these analytical results have only been shown to be valid for the specific form of the skill-building equation (i.e., the state equation) used. Therefore, it behooves us to examine some alternative forms of the skill-building equation and study the impact of these different forms on the results presented earlier. Unfortunately, however, as the skill-building forms become more complex, the model becomes analytically intractable. We, therefore, conducted these examinations numerically on several alternative skill-building forms with nonlinear specifications. In all of these experiments, the focus was to test the main conclusion of the previous section, namely that the use of open source for training increases in response to an increase in labor market pressure. Our finding in these experiments was that this conclusion is upheld in all the different forms of skill-building that were tested. Thus it appears that the use of open source participation as a means of compensating programmers in tight labor markets is a quite a robust result, and is fairly immune to the specific form of the skill-building process.

Future Work

A direct extension to the current study is to consider the impact of open source on the average level of skill in the labor market for programmers. The presence of open source provides a training mechanism that incentivizes firms to increase training (not just because it is free but also because they can leverage training from open source to pay lower wages) thereby increasing average skill levels in the labor market. An analysis of the extent of this impact will provide a sound economic basis to help answer policy questions like how much should regulators intervene in stimulating open source activity.

At a broader level, the present study could also motivate future studies that focus on an empirical investigation of related issues. For example, what kinds of firms benefit more from their programmers learning through open source project participation? Another area is the creation of a match between a particular closed source and available open source projects and investigating how this match affects outcomes (productivity, value) in a firm. It is also important to compare the efficacy of open source learning with alternative informal training opportunities like employee mentorship programs or using communities of practice. Such issues are quite relevant as firms experiment with these other informal learning mech-

anisms. Such efforts are relatively common today and commercial platforms to develop and maintain communities of practice exist (<http://www.icohere.com/webcommunities.htm>). One can also study the relative benefit of informal training though open source *vis-à-vis* that of formal classroom training and focus on the design of a comprehensive training program with different kinds of training inputs.

References

- Arrow, K. J. 1962. "The Economic Implications of Learning by Doing," *The Review of Economic Studies* (29:3), pp. 155-173.
- Baldwin-Evans, K. 2006. "Key Steps to Implementing a Successful Blended Learning Strategy," *Industrial and Commercial Training* (38:3), pp. 156-163.
- Baldwin-Evans, K. 2007. "The Future of Organizational Learning," *Industrial and Commercial Training* (39:6), pp. 299-306.
- Becker, G. S. 1962. "Investment in Human Capital: A Theoretical Analysis," *Journal of Political Economy* (40), pp. 9-49.
- Becker, G. S., and Murphy, K. M. 1992. "The Division of Labor, Coordination Costs, and Knowledge," *The Quarterly Journal of Economics* (107:4), pp. 1137-1160.
- Berg, S. A., and Chyung, S. Y. 2008. "Factors that Influence Informal Learning in the Workplace," *Journal of Workplace Learning* (20:4), pp. 229-244.
- Bishop, J. H. 1991. "On-the-Job Training of New Hires," in *Market Failure in Training?*, D. Stern and J. M. M. Ritzen (eds.), New York: Springer-Verlag, pp. 61-98.
- Blaug, M. 1976. "The Empirical Status of Human Capital Theory: A Slightly Jaundiced Survey," *Journal of Economic Literature* (16), pp. 827-855.
- Boh, W. F., Slaughter, S. A., and Espinosa, J. A. 2007. "Learning from Experience in Software Development: A Multilevel Analysis," *Management Science* (53:8), pp. 1315-1331.
- Bolton, P., and Dewatripont, M. 1994. "The Firm as a Communication Network," *The Quarterly Journal of Economics* (109:4), pp. 809-839.
- Boucekkine, R., and Crifo, P. 2008. "Human Capital Accumulation and the Transition from Specialization to Multitasking," *Macroeconomic Dynamics* (12:3), pp. 320-344.
- Cable, D., and Parsons, C. 2001. "Socialization Tactics and Person-Organization Fit," *Personnel Psychology* (54:1), pp. 1-23.
- Cartensen, V. 2002. "The From-Tayloristic-to-Holistic-Organization Model from an Empirical Perspective," Discussion Paper No. 256, University of Hanover.
- Center for Workforce Development. 1998. *The Teaching Firm: Where Productive Work and Knowledge Converge*, Newton, MA: Education Development Center, Inc.
- Conlon, T. J. 2004. "A Review of Informal Learning Literature, Theory and Implications for Practice in Developing Global Professional Competence," *Journal of European Industrial Training* (28:2/3/4), pp. 283-295.
- Garavan, T. N. 1987. "Promoting Natural Learning Activities Within the Organisation," *Journal of European Industrial Training* (11:7), pp. 18-22.
- Garvin, D. A., Edmondson, A. C., and Gino, F. 2008. "Is Yours a Learning Organization?," *Harvard Business Review* (86:3), pp. 109-116.

- Gibbs, M., Levenson, A., and Zoghi, C. 2010. "Why Are Jobs Designed the Way They Are?," *Research in Labor Economics* (30), pp. 107-154.
- Hann, I. H., Roberts, J., Slaughter, S., and Fielding, R. 2006. "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects," *Management Science* (52:7), pp. 984-999.
- Harvard Management Update. 2000. "Creating an Informal Learning Organization," July 1.
- 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), pp. 1159-1177.
- Holmstrom, B., and Milgrom, P. 1991. "Multi-Task Principal-Agent Analyses: Incentive Contracts, Asset Ownership and Job Design," *Journal of Law, Economics and Organization* (7), pp. 24-52.
- 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), pp. 263-271.
- Lakhani, K. R., and von Hippel, E. 2003. "How Open Source Software Works: 'Free' User-to-User Assistance," *Research Policy* (32), pp. 923-943.
- Lakhani, K. R., and Wolf, R. G. 2005. *Perspectives on Free and Open Source Software*, Cambridge, MA: MIT Press.
- Larson, B. 1991. "Informal Workplace Learning and Partner Relationships Among Paramedics in the Pre-hospitals Settings," Unpublished Doctoral Dissertation, Teachers College, Columbia University, New York, NY.
- Lee, G. K., and Cole, R. E. 2003. "From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development," *Organization Science* (14:6), pp. 633-649.
- Lindbeck, A., and Snower, D. J. 1996. "Reorganization of Firms and Labor-Market Inequality," *The American Economic Review* (86:2), pp. 315-321.
- Lindbeck, A., and Snower, D. J. 2000. "Multitask Learning and the Reorganization of Work: From Tayloristic to Holistic Organization," *Journal of Labor Economics* (18:3), pp. 353-376.
- Lindbeck, A., and Snower, D. J. 2001. "Centralized Bargaining and Reorganized Work: Are They Compatible?," *European Economic Review* (45:10), pp. 1851-1875.
- Mehra, A., Dewan, R., and Freimer, M. 2011. "Firms as Incubators of Open Source Software," *Information Systems Research* (22:1), pp. 22-38.
- Meizner, A., Healy, A., West, D., Conolly, T., Glott, R., Sowe, S. K., and Weller, M. J. 2008. FLOSS-Like Education Transfer Report," Working Package Number 05, FLOSSCom.net.
- Menon, T., and Pfeffer, J. 2003. "Valuing Internal vs. External Knowledge: Explaining the Preference for Outsiders," *Management Science* (49:4), pp. 497-513.
- Merriam, S., and Caffarella, R. 1991. *Learning in Adulthood*, San Francisco: Jossey-Bass.
- Moody, G. 2002. *Rebel Code: Inside Linux and the Open Source Revolution*, New York: Perseus Press.
- Pautrel, X. 2004. "A Note on Multitask Learning and the Reorganization of Work," *Economics Bulletin* (10:5), pp. 1-6.
- Raymond, E. S. 2001. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Sebastapol, CA: O'Reilly Publications.
- Ryu, C., Kim, Y. J., Chaudhury, A., and Rao, H. R. 2005. "Knowledge Acquisition via Three Learning Process in Enterprise Information Portals: Learning-by-Investment, Learning-by-Doing, and Learning-from-Others," *MIS Quarterly* (29:2), pp. 245-278.
- Schilling, M. A., Vidal, P., Polyhart, R. E., and Marangoni, A. 2003. "Learning by Doing Something Else: Variation, Relatedness, and the Learning Curve," *Management Science* (49:1), pp. 39-56.
- Sethi, S. P., and Thompson, G. L. 2000. *Optimal Control Theory Applications to Management Science and Economics*, Norwell, MA: Kluwer Academic Publishers.
- Sheshinski, E. 1968. "On the Individual's Lifetime Allocation Between Education and Work," *Metroeconomica* (20:1), pp. 42-49.
- 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*, (http://ec.europa.eu/enterprise/sectors/ict/files/2006-11-20-flossimpact_en.pdf)
- von Krogh, G., Spaeth, S., and Lakhani, K. R. 2003. "Community, Joining, and Specialization in Open Source Software Innovation: A Case Study," *Research Policy* (32:7), pp. 1217-1241.
- Wayner, P. 2000. *Free For All: How Linux and the Free Software Movement Undercut the High-Tech Titans*, New York: HarperBusiness.
- Yang, X., and Borland, J. 1991. "A Microeconomic Mechanism for Economic Growth," *Journal of Political Economy* (99:3), pp. 460-482.

About the Authors

Amit Mehra is an assistant professor in Information Systems at the Indian School of Business. He received his Ph.D. from the University of Rochester. He draws upon game theory, agency theory, and econometric tools to research topics in software product development, pricing and launch strategies, retailing on the Internet, human capital development for IT firms, and interfirm outsourcing relationships. He has published in *Information Systems Research* and presented at top Information Systems and Marketing conferences.

Vijay S. Mookerjee is the Charles and Nancy Davidson Distinguished Professor of Information Systems in the School of Management, University of Texas at Dallas. He holds a Ph.D. in Management, with a major in MIS, from Purdue University. His current research interests include social networks, optimal software development methodologies, storage and cache management, content delivery systems, and the economic design of expert systems and machine learning systems. He has published in and has articles forthcoming in several archival Information Systems, Computer Science, and Operations Research journals. He serves (or has served on) the editorial board of *Management Science*, *Information Systems Research*, *INFORMS Journal on Computing*, *Operations Research*, *Decision Support Systems*, *Information Technology and Management*, and *Journal of Database Management*.

HUMAN CAPITAL DEVELOPMENT FOR PROGRAMMERS USING OPEN SOURCE SOFTWARE

Amit Mehra

Indian School of Business, Hyderabad, INDIA {Amit_Mehra@isb.edu}

Vijay Mookerjee

School of Management, University of Texas at Dallas, Richardson, TX 75080 U.S.A. {vijaym@utdallas.edu}

Appendix

Proofs of Lemmas and Propositions

Proof of Lemma 1

The maximum value of w needed to satisfy the IR constraint of the programmer (called w_{IR}) is obtained from $w_{IR}T = M_0 - ax(T)|_{v=0}$. Here $x(T)|_{v=0}$ represents the smallest possible skill level with which the programmer ends the contract. This skill level occurs when $v = 0$ is chosen throughout the contract duration. Next, note that the adjoint equation for the co-state variable $\mu(t)$ is $\dot{\mu} = -\frac{\partial H}{\partial v} = 0 \Rightarrow \mu(t) = \text{constant} \forall t$.

Thus μ must take the same value throughout the contract duration. If $\mu > 1$, then $\frac{\partial H}{\partial w} > 0$ implying that w must be the maximum possible and hence the optimal value of $w > w_{IR}$, since the domain of the wage premium w is R^+ . However, such a choice of w results in a smaller net value for the firm than if the firm chose $w = w_{IR}$ since paying wages is a cost to the firm and has no impact on programmer productivity. This is a contradiction because a suboptimal value of w gives a better solution to the firm. Hence it must be that $\mu \leq 1$. If $\mu < 1$, then $\frac{\partial H}{\partial w} < 0 \Rightarrow w = 0$.

On the other hand, if $\mu = 1$, then $\frac{\partial H}{\partial w} = 0 \Rightarrow w$ can take any possible positive value (singular solution).

Proof of Lemma 2

We represent $\frac{\partial H}{\partial v}$ by z . Thus, using Equation 4, we can write

$$\lambda = \frac{z + Kx}{\alpha} - a\mu \tag{6}$$

Next, taking the derivative of both sides of Equation 4 with respect to t , we have

$$\dot{z} = -K\dot{x} + \dot{\lambda}\alpha + \lambda(\alpha_t + \alpha_x\dot{x}) + a\mu(\alpha_t + \alpha_x\dot{x})$$

Substituting in the value of \dot{x} from Equation 1, and that of $\dot{\lambda}$ from the adjoint equation, $\dot{\lambda} = -\frac{\partial H}{\partial x} = -(K(1-\nu) + \lambda(\nu\alpha_x + \delta_x - \beta_x) + a\mu(\nu\alpha_x + \delta_x - \beta_x))$, we can rewrite the above equation as

$$\lambda = \frac{K(\beta - \delta - \alpha) - \dot{z}}{\alpha_x(\beta - \delta) - \alpha_t - \alpha(\beta_x - \delta_x)} \tag{7}$$

Equating the values of λ from equations 6 and 7, we obtain

$$\dot{z} = z \left(-\frac{\alpha_x(\beta - \delta) - \alpha_t - \alpha(\beta_x - \delta_x)}{\alpha} \right) + \left(K(\beta - \delta - \alpha) - \frac{Kx(\alpha_x(\beta - \delta) - \alpha_t - \alpha(\beta_x - \delta_x))}{\alpha} \right) \tag{8}$$

Note that the coefficient of z as well as the term independent of z are purely functions of x and are independent of the controls or t . Hence, we represent these two terms by $-g(x,t)$ and $h(x,t)$, respectively.

Now suppose that $z = 0$ at two different instants τ_0 and τ_1 , where $\tau_0 < \tau_1$, without loss of generality. We consider an instant t such that $\tau_0 < t < \tau_1$ and $z(t) \neq 0$. The general solution of the differential equation 8 is

$$z(t) \exp \int g(x,\sigma)d\sigma = \int \exp \int g(x,\sigma)d\sigma h(x,s)ds + C$$

where C is the constant of integration. Using this equation we can write the solution between t and τ_1 as

$$z(\tau_1) \exp \int_0^{\tau_1} g(x,\sigma)d\sigma - z(t) \exp \int_0^t g(x,\sigma)d\sigma = \int_0^{\tau_1} (h(x,s)) \exp \int_0^s g(x,\sigma)d\sigma ds$$

Since $z(\tau_1) = 0$, this can be rewritten as

$$z(t) \exp \int_0^t g(x,\sigma)d\sigma = - \int_0^{\tau_1} (h(x,s)) \exp \int_0^s g(x,\sigma)d\sigma ds \tag{9}$$

Similar to above, we consider the solution of the differential equation 8 between τ_0 and t . This gives us

$$z(t) \exp \int_0^t g(x,\sigma)d\sigma = - \int_{\tau_0}^t (h(x,s)) \exp \int_0^s g(x,\sigma)d\sigma ds \tag{10}$$

From equations 9 and 10, we note that $z(t)$ must have two different values at t . However, equation 8 implies that $z(t)$ must be a continuous function. Hence, it cannot have two values at any point. Hence our supposition that $z = 0$ at two non-contiguous instants must be incorrect. Since z is continuous in t , it can be either positive or negative prior to and after when it becomes zero. If $z > 0$, $\nu = 1$ and if $z < 0$, $\nu = 0$. This gives us the statement of the lemma.

Proof of Proposition 1

Statement A of the proposition follows from Lemma 2.

For Statement B, note that $\frac{\partial H}{\partial \nu} = 0$ and $\frac{d}{dt} \frac{\partial H}{\partial \nu} = 0$ are required for a singular solution since the partial derivative of the Hamiltonian with respect to the control must be zero and must continue to remain to be zero for some interval. Also, $\dot{\lambda} = -\frac{\partial H}{\partial x}$ is the standard adjoint condition. These three equations simplify to the following:

$$-Kx + \alpha(\lambda + a\mu) = 0 \tag{11}$$

$$-K\alpha + K(\beta - \delta) - (\lambda + a\mu)(\alpha_x(\beta - \delta) - \alpha_t - \alpha(\beta_x - \delta_x)) = 0 \tag{12}$$

$$\dot{\lambda} = -(K(1 - \nu) + \lambda(\nu\alpha_x + \delta_x - \beta_x) + a\mu(\nu\alpha_x + \delta_x - \beta_x)) \tag{13}$$

Simultaneously solving equations 11 and 12 we obtain the solutions for x and λ , whereupon we can find $\dot{\lambda}$. Substituting in values of λ , $\dot{\lambda}$, and x in Equation 13, we obtain the value of ν for the singular solution.

For Statement C, note that for $\mu = 0$, the IR constraint is not binding. Further, $\lambda(T) = 0$ since the firm does not have a salvage value of programmer skills. From Equation (4), we have $\frac{\partial H}{\partial v}|_{v=T} = -Kx < 0$. From Lemma 2, we know that $\frac{\partial H}{\partial v}$ does not change sign in \hat{t} to T . Hence $\nu = 0$ in this interval. When $0 \leq \mu \leq 1$ (i.e., when IR binds), $\frac{\partial H}{\partial v}$ is either positive or negative in this interval. Correspondingly, $\nu = 1$ or $\nu = 0$.

Proof of Lemma 3

(A) In the proof of Lemma 2, from Equation 9, we see that the sign of the Hamiltonian, $z(t)$, is opposite to the sign of function $h(x, t)$ for $t < \bar{t}$. With the simplified state equation, we have $h(x, t) = \frac{-c_1c_2K - c_2^2K + 2c_2(c_4 - c_3)Kx + c_3(c_4 - c_3)Kx^2}{c_2 + c_3x}$. Using this we can easily show that $x(t) < \bar{x} \Rightarrow z(t) > 0 \Rightarrow \nu = 1$ and $x(t) > \bar{x} \Rightarrow z(t) < 0 \Rightarrow \nu = 0$. By Lemma 2, it must be that $z(t)$ has the same sign in the interval 0 to \bar{t} . Accordingly, it is sufficient to check if $x_0 < \bar{x}$. If that is true then $\nu = 1$ and if not then $\nu = 0$.

(B) Applying the method outlined in proof of Proposition 1, we find the values of ν , x , and λ in the singular region ($\bar{t} < t \leq \hat{t}$). These values are indicated by $\bar{\nu}$, \bar{x} , and $\bar{\lambda}$, respectively.

Note that $\bar{\nu}$ and \bar{x} are independent of μ . Consequently, the optimal values of ν in the pre-singular and the singular region are unaffected by whether the IR constraint is binding or not.

(C) In the region $\hat{t} \leq t \leq T$, we know that $\nu = 0$ when the IR constraint is not binding (Proposition 1).

Suppose that $\nu = 1$ is a solution when the IR constraint is binding. The corresponding Hamiltonian is $H_1 = -w + \lambda(c_1 + c_2 + c_3x - c_4x) + \mu(w + a(c_1 + c_2 + c_3x - c_4x))$. Using the adjoint equation $\dot{\lambda} = -\frac{\partial H_1}{\partial x}$. This implies that $\dot{\lambda} = (c_4 - c_3)(\lambda + a\mu)$. Note that the sign of $\dot{\lambda}$ crucially depends upon the value of λ at $t = \hat{t}$ (i.e., $\bar{\lambda}$). Also note that λ must be equal to 0 at $t = T$ since the firm has no salvage value for the skills of the programmer at $t = T$. If $\bar{\lambda} > 0$ or if $\bar{\lambda} < -a\mu_0$, then this is impossible (since then λ either monotonically increases from a positive value or monotonically decreases from a negative value). However this may be possible when $\bar{\lambda} < 0$ and $\bar{\lambda} + a\mu > 0$ (now λ can increase monotonically from a negative value and reach zero). It is easy to verify that the second condition always holds under requirements imposed on the parameters c_1, c_2, c_3 , and c_4 . The first condition requires $\mu > \frac{(c_1c_3 + c_2c_4 - \sqrt{c_2(c_4 - c_3)}(c_1c_3 + c_2c_4))K}{ac_3(c_1c_3 + c_2c_4)}$. Note from the discussion preceding Lemma 1 that $\mu \leq 1$. This requires $K \leq \frac{ac_3(c_1c_3 + c_2c_4)}{c_1c_3 + c_2c_4 - \sqrt{c_2(c_4 - c_3)}(c_1c_3 + c_2c_4)}$. This is ruled out due to our restriction on marginal productivity of programmers. Hence, $\nu = 1$ cannot be a solution in this region when the IR constraint is binding.

Hence, $\nu = 0$, the only remaining possibility is the solution.

Proof of Proposition 2

The general solution to equation 5 is $x = \frac{c_1 + c_2\nu}{c_4 - c_2\nu} + \exp^{-(c_4 - c_3)\nu} C$, where C is the constant of integration. Using $x = x_0$ at $t = 0$ we obtain C .

We know that $x = \bar{x}$ at $t = \bar{t}$. Using this relation, we get

$$\hat{t} = \frac{\text{Log} \left[\frac{(c_1 c_3 (c_3 - c_4) + c_2 (c_3 - c_4) c_4 + \sqrt{c_2 (c_4 - c_3) (c_1 c_3 + c_2 c_4) (-c_4 + c_3 v)}) (c_1 + c_2 v - c_4 x_0 + c_3 v x_0)}{(c_1 c_3 + c_2 c_4) (c_1 (c_3 - c_4) + c_2 (c_4 - 2c_4 v + c_3 v^2))} \right]}{c_4 - c_3 v}$$

This expression is independent of μ . Hence it is applicable irrespective of whether the IR constraint is binding or not.

From Lemma 3, we know that $v(t) = 0 \forall t \in [\hat{t}, T]$. Now we write the differential equation for λ in this region. Using the adjoint equation, $\dot{\lambda} = -\frac{\partial H}{\partial x}$, this is

$$\dot{\lambda} = ac_4 \mu + c_4 \lambda - K \tag{14}$$

The general solution for this equation is $\lambda = -\frac{ac_4 \mu - K}{c_4} + \exp^{c_4 t} C$, where C is the constant of integration and t is transformed to the scale $[0, T - \hat{t}]$. Since, there is no salvage value of skills for the firm, we have $\lambda = 0$ at $t = T - \hat{t}$. Using this we obtain the expression for λ as a function of \hat{t} . This expression is used to find λ at $t = 0$ which is then equated with $\bar{\lambda}$ that we obtained in the proof for Lemma 3. This

enables us to get $\hat{t} = T - \frac{\text{Log} \left[\frac{(-c_1 c_3 (c_4 - c_3) - c_4 (c_2 (c_4 - c_3) + \sqrt{c_2 (c_4 - c_3) (c_1 c_3 + c_2 c_4)})) (K - ac_4 \mu)}{-(c_4 - c_3) (c_1 c_3 + c_2 c_4 - c_1 c_4) K} \right]}{c_4}$. Note that $\bar{v} > 0$ requires

$c_2 > \frac{c_1 (c_4 - c_3)}{c_4}$. This condition makes the denominator of the *Log* term negative. Thus we can have a real value of \hat{t} only when the numerator of the *Log* term is also negative. This will happen only when $K > ac_4 \mu$. From discussion preceding Lemma 1 we know that $\mu \leq 1$. Note that $K > ac_4$ because of our restriction on marginal productivity of programmers and the requirement on parameters due to $\bar{v} > 0$. Hence \hat{t} is real valued. Thus, since $K > ac_4 \mu$, $\frac{\partial \hat{t}}{\partial \mu} = \frac{a}{K - ac_4 \mu} > 0$, which is what we needed to establish.

Proof of Proposition 3

Consider a situation where the IR constraint binds but the firm does not pay any wage premium to the programmer ($w = 0$). In this situation, it must be that $ax(T) = M_0$. Using equation 5 and the boundary conditions $x = \bar{x}$ at $t = 0$ and $x = \frac{M_0}{a}$ at $t = T - \hat{t}$, and using the fact that $v = 0$

in between \hat{t} and T , we find an expression for $\hat{t}_{IRB} = T + \frac{\text{Log} \left[\frac{(c_1 c_3 (c_3 - c_4) - c_4 (c_2 (c_4 - c_3) + \sqrt{c_2 (c_4 - c_3) (c_1 c_3 + c_2 c_4)})) (ac_1 - c_4 M_0)}{a (c_1 c_3 + c_2 c_4) (c_1 c_3 + c_2 c_4 - c_1 c_4)} \right]}{c_4}$.

Equating this expression with that \hat{t} from proof of Proposition 2, we get an expression for μ in terms of M_0 . It is algebraically tedious, but easy to show that μ is increasing in M_0 . Using this property of μ , we can show that $0 \leq \mu \leq 1$ for $M_1 \leq M_0 \leq M_2$. Note that when $\mu = 1$, $\frac{\partial H}{\partial w} < 0$ and so $w = 0$. However, if $M_0 > M_2$ then $\mu = 1 \Rightarrow \frac{\partial H}{\partial w} = 0$, and hence the firm can pay a positive wage w .

Proof of Proposition 4

Part 1

Suppose $x_0 > \bar{x}$. From Lemma 3, we know that $\frac{\partial H}{\partial v} |_{t=T} < 0$. Now either $\frac{\partial H}{\partial v} < 0 \forall t \in [0, T]$ implying $v(t) = 0$, or else there must be some τ such that $0 < \tau < T$ where $\frac{\partial H}{\partial v} |_{t=\tau} = 0$. Suppose that such a τ exists. Note that since the singular solution is ruled out, $\frac{\partial H}{\partial v}$ can be 0 at most

at some instant but not in an interval. Then as per Lemma 2, $v(t) = 0 \forall t \in [\tau, T]$. Further, as per Lemma 3, $v(t) = 0 \forall t \in [0, \tau]$. Hence, $v(t) = 0$ throughout the contract duration is the only solution. This also shows that the firm never uses the option of training the programmers even if the IR constraint binds.

Part 2

Suppose $x_0 < \bar{x}$ and that there exists some τ such that $0 < \tau < T$ where $\frac{\partial H}{\partial v}|_{t=\tau} = 0$. As in Part 1, we know that singular solution is not possible,

hence $\frac{\partial H}{\partial v}$ can be 0 at most at some instant but not in an interval. Then using Lemma 3, we know that $v(t) = 1$ for $t \in [0, \tau]$ and $v(t) = 0$ for

$t \in [\tau, T]$. To find τ , we first consider the region $\tau < t < T$. Using the adjoint equation, $\dot{\lambda} = -\frac{\partial H}{\partial x}$ and the end condition $\lambda = 0$ at $t = T$, we work

out $\lambda(t) = \frac{K - ac_4\mu}{c_4} - \frac{\exp^{-c_4(T-t+\tau)}(K - ac_4\mu)}{c_4}$. We represent x at $t = \tau$ by x_τ . Using this as the boundary condition, and using the state

equation 5, we find $x(t) = \frac{\exp^{c_4 t}(c_4 x_\tau - c_1) + c_1}{c_4}$. Using the expressions for $x(t)$ and $\lambda(t)$, we can work out the expression for $\frac{\partial H}{\partial v}$. Now

utilizing $\frac{\partial H}{\partial v}|_{t=\tau} = 0$, we find $x_\tau = \frac{c_2 \exp^{c_4 \tau}((-1 + \exp^{c_4(T-\tau)})K + ac_4\mu)}{(c_4 - c_3) \exp^{c_4 T} K + c_3 \exp^{c_4 \tau} (K - ac_4\mu)}$.

Now we consider the region $0 < t < \tau$. Using the state equation 5 and the boundary condition $x(0) = x_0$, we can work out the expression for $x(t)$. Now, τ can be obtained from solving

$$x(t)|_{t=\tau} = x_\tau \tag{15}$$

This is the equation in the statement of the proposition.

To show that τ is unique, we take the derivative of x_τ w.r.t τ . This is easily seen to be negative. Next we take the derivative of $x(t)|_{t=\tau}$ w.r.t τ and this expression shows that its minimum value occurs at $x_0 = \bar{x}$. This minimum value of the slope is found to be positive. Thus the expressions on the left hand and right hand side of the equation above have opposite signs w.r.t τ . Thus τ must be unique.

Finally, if $\tau < 0$, then $v = 0$ for $0 \leq t \leq T$.

To see how τ behaves as the IR constraint binds, we rewrite equation 15 as $x(t)|_{t=\tau} - x_\tau = 0$ and represent the LHS this equation by f . Then taking the derivative w.r.t μ we have $\frac{df}{d\mu} + \frac{df}{d\tau} \frac{d\tau}{d\mu} = 0$. It is easy to show that $\frac{df}{d\mu} < 0$ and $\frac{df}{d\tau} > 0$. Thus, it must be that $\frac{d\tau}{d\mu} > 0$. Clearly as μ increases, τ also increases leading to a greater training period at the beginning of the contract duration. This leads to higher skills at the conclusion of training, thus enabling the firm to pay lower wages. However, we know that $\mu \leq 1$. Hence, once μ reaches 1, no further extension in τ is possible and any shortfall in IR constraint must then be paid through a wage premium.