

# Spence Revisited - Signalling with Externality: The Case of Open Source Programming<sup>1</sup>

Mikko Leppämäki<sup>2</sup>  
FPPE & RUESG, University of Helsinki  
&  
Mikko Mustonen<sup>3</sup>  
FPPE, University of Helsinki

Department of Economics, University of Helsinki  
Discussion Papers, No. 558:2003  
ISSN 1459-3696  
ISBN 952-10-0699-4  
2.4.2003

---

<sup>1</sup>We thank seminar participants at RUESG workshop in October 2002 for their comments.

<sup>2</sup>**Address:** FPPE, P.O. Box 54, FIN-00014 University of Helsinki, Finland. e-mail: mikko.leppamaki@helsinki.fi

<sup>3</sup>**Address:** Department of Economics, P.O. Box 54, FIN-00014 University of Helsinki, Finland. e-mail: mikko.mustonen@helsinki.fi

## **Abstract**

This paper considers job market signalling in the software industry where individual programmers engage themselves in open source software projects in order to signal their skills to commercial software companies. We provide a novel signalling game where signalling in the labor market has a (positive or negative) product market externality effect due to the appearance of free open source programs. We show how the least cost separating equilibrium of the signalling game is affected by product market externalities, and how this is reflected in the quality of goods supplied in the market. Spence's (1973,1974) original case turns out to be a special case of ours when market externalities are absent. We also show that in the case of strong positive market externalities, i.e. with complementarity between OS and commercial programs, the separation of programmers' types may be impossible.

**JEL Classification numbers:** D23, D82.

**Keywords:** Signalling, Labor market, Open source software.

# 1 Introduction

This paper is inspired by two observations from the open source software environment and software industry in general. It is widely documented that individual programmers engage in developing software in open source environments that provides no immediate direct (monetary) payoffs for their programming effort<sup>4</sup>. Why do they get involved in such ventures? Equally well, commercial software companies do support and subsidize such open source communities in various ways.<sup>5</sup> What is the motivation behind firms' behavior given that these communities may potentially be even harmful as providing free open source programs that compete with their own commercial copyright programs?

The main idea of this paper is to show within the formal model that indeed such a behavior may be fully rational. It can be explained by the incentives of independent programmers to participate in open source projects in order to signal their innate ability (productivity or programming ability) to commercial software companies. That is, we base our work on skill signalling approach (c.f. Lerner and Tirole 2002), and develop a simple labor market skill signalling model that extends Spence's (1973, 1974) seminal work into a case where the signalling activity itself has an externality effect that is coming through product markets. Already Spence discussed the possibility of education increasing productivity, but the case where the labor market signalling has an externality effect via the product market is, to the best of our knowledge, novel. In fact, our approach is general enough to incorporate both positive and negative product market externalities. That is, we can handle both complementary and substitute products, and consequently this means that the receiver of signal (i.e. the commercial software company) may benefit or be hurt due to it. As examples, Openoffice is likely to be a complement to commercial operating systems like Windows, whereas Linux is a substitute for them. Moreover, in our case the signalling activity as such is not social waste since it will eventually realize in a free product (software) that consumers value.

In our model, there are three types of players: programmers, firms and consumers, who will interact in two types of market: labor market and product market. Programmers engage themselves in open source programming projects. The peculiar licensing structure of open source - one example is the GPL (copyleft) license (GNU 2000a,2000b) - preserves the link between

---

<sup>4</sup>For descriptions of open source, see Raymond 1998, Browne 1999, GNU 2000a, 2000b, Kogut and Metiu 2001 and Lerner and Tirole 2002.

<sup>5</sup>See IBM News 2001, Collab.net 2001, Hann et al. 2002, Mustonen 2002b.

the originator and the program code while making the program public and freely available. We analyze programmers' (who differ in their innate ability) signalling by open source programming and software companies' hiring and derive a PBE of this signalling game.

We start by providing a benchmark result which shows that the case considered in Spence's classical work will turn out to be a special case of ours when the signalling externality effect (that we choose to emphasize) is not present because programs are independent. Then, not surprisingly, we show that in the case of substitutes the programming credential needed to separate the "good type" from the "bad type" is lower than in the case of independent programs. This we can interpret as indicating that in the case of substitutes the commercial software companies are willing to provide lower wages (due to negative product market externality), and thus in the equilibrium the least cost separating programming credential is lower as well. In the end this means that in terms of quality (as lines of code transforming into the functionality of a program), the substitute open source programs have lower quality than the independent programs. Similarly, we can show that in the case complementary programs, the equilibrium programming credential will be higher than in the case of independent programs, and thus the resulting free open source programs will have higher quality (more lines of code, and thus functionality). In the case of complementary programs, it may interestingly even be the case that it is impossible for good types to separate from bad ones given that the product externality effect is strong enough, and thus one has a pooling equilibrium.

In another version of the basic model we discuss the role of an open source project as a screening device. In that case we assume there exists a high enough outside option wage that implies that the commercial software companies cannot even exist without screening of workers. It is shown that they may indeed be willing to subsidize open source projects in order to be able to screen workers to enjoy non-negative profits. At the end, we discuss some welfare issues and point out that there may be a conflict of interest, since the individual programmer (and the software company) does not internalize the beneficial effects that are coming via the consumer's surplus.

It is interesting to contrast our theoretical model with what happens in software industry in practise. In reality, open source software projects do have home pages on internet where they in fact post merit-based ranking lists of the most important contributors (Hann et al. 2002, Linux-PAM 2003). They could even be called as "hall of fames". The fact that a programmer is at the top of the list implies that he/she has contributed to the project in a

major way by providing important or even crucial pieces of code (parts of a program). We interpret this as being a *programming credential* that signals the programmers' innate abilities. That is, for a more able programmer it is easier to get the top position at the merit based list.

In their recent paper Hann, Roberts and Slaughter (2002) provide first empirical evidence of economic incentives of individual programmers within the Apache web-server open source project.<sup>6</sup> Their empirical results confirm the existence of economic returns; participation per se as measured by the numbers of contributions made does not lead to wage increases, but a higher status in a merit-based ranking does lead to significantly higher wages. A higher status in a merit-based ranking list is a credible signal of the productive capacity of a programmer. Thus their work gives support for the delayed returns argument - motivation for participation is skill signalling<sup>7</sup>. Commercial software companies are willing to pay for high wages for the top performers i.e. they interpret a high position at the merit-based ranking list as a signal of high innate productivity.

In the open source literature one can distinguish two lines of literature; incentive based approach (Johnson 2002, Bessen 2001) and market analysis (Lerner and Tirole 2002, Mustonen 2002a). Schiff (2002) provides a survey of the early literature. This paper combines the two lines of arguments by developing a single model that incorporates the idea of signalling in labor market with externality in product market due to appearance of free open source programs, and where moreover the product market effect may be harmful or beneficial to the receiver of signal. The novelty of this paper is that introduces a new way how the signalling may work and it also provides a new reason for why firms may participate in and subsidize open source projects.

The structure of the rest of the paper is as follows. In next section we present the model and the main analysis is carried out in section 3. In section 4 we show that commercial software companies do have incentives to support open source projects in order to screen their work force. Section 5 focuses on the welfare issues and section 6 concludes.

---

<sup>6</sup>Apache is used in 63% of the world's over 100 million web servers (Netcraft 2001).

<sup>7</sup>For discussions on delayed returns, we refer to Dasgupta and David (1987,1994), Dewatripont, Jewitt and Tirole (1999) and Stern (1999).

## 2 The Model

In this section we set up the model by first describing the parties involved and the markets where they meet and interact. On purpose, we follow the exact notation of Spence (1974) as closely as possible. In particular, we want to emphasize that the case considered in Spence's classical article will turn out to be a special case of ours when the signalling externality effect (that we choose to emphasize) is not present.

### 2.1 Players: programmers, software companies and consumers

We consider a model with three types players who will interact in two types of markets. We have *workers* called as programmers who develop computer software either within the open source projects or within *firms* called as commercial software companies. Finally there are *consumers* who value available computer software. The two markets we consider are labor markets where computer programmers and software companies interact and products markets where consumers decide whether they buy a commercially produced software or acquire an open source software for free.

### 2.2 Labor Market: programmers, software companies and the signalling externality $k$

We assume there exists two types of programmers, who differ only in their innate non-verifiable ability. It is assumed that the high productivity programmers ("good type") have a marginal productivity of 2 (units of program code per unit of time) and the low productivity programmers ("bad type") have a marginal productivity of 1, and that the share of good types is equal to  $q_1$ , and thus  $(1 - q_1)$  stands for the share of low productivity programmers.

In order to separate from the low productivity programmer, the high productivity programmer may engage himself in an open source programming project. If an individual programmer's contribution to the open source programming project is large enough, his name will appear in the project's 'top contributors' or 'hall of fame' list, which is public information.<sup>8</sup> We call this a *programming credential* and label it by  $y$ .

Notice that  $y$  bears a similarity to Spence's term education level in the sense that a given level is more easily attainable for a high productivity pro-

---

<sup>8</sup>In reality such information is often posted on the open source projects' homepages in the internet. See Linux -PAM (2003) and consult Hann et al (2002).

grammer. Interestingly, within the context of open source programming, the programming credential may have an externality effect. The programmer's effort to separate (i.e. to signal his ability) is directed to writing program code to the open source program. We define that for a programming credential level of  $y$ , the programmer's effort is manifested in  $|k|y$  lines of program code that affects the consumer market for software.

We interpret  $k$  as an externality effect from skill signalling because the outcome of the open source programming project is freely available to consumers. Notice that the externality we describe is novel and differs from the possible productivity effect that already Spence discussed. He acknowledged that acquiring education as such may increase the employee's productivity, so that the productivity of good type after attaining education level  $y$  becomes  $2 + y/4$ . In contrast to that, here the productivity of employees remains unchanged but the product market, where the commercial software company is present, is affected via a positive or negative externality effect as described above.

Finally, the programmers' utilities are assumed to depend on the wage and the disutility of attaining the credential,

$$U_G = w - \frac{y}{2}, U_B = w - y.$$

In above  $G$  refers to "good", high productivity type and  $B$  to "bad", low productivity type each attaining programming credential  $y$  and earning the wage  $w$ .

### 2.3 Product market: consumers and software companies

It is assumed that there exists a market niche where a firm supplies its commercial software to consumers. The commercial program supplied by the firm is developed by hiring a programmer, and in order to simplify we assume that the size of the firm's programming project is equal to one. That is, each firm hires one programmer and develops one program. If the firm hires a good programmer, the size of the program (as lines of code) is  $S = 2$  and if it hires a bad one, the size is  $S = 1$ . When the firm does not screen programmers ex ante, the expected size of the program thus is  $S = 2q_1 + (1 - q_1) = q_1 + 1$ .

We assume that there are  $M$  consumers, each buying at most one unit of either program<sup>9</sup>. They assess the programs available on the market and

---

<sup>9</sup>The market analysis draws on Koboldt (1995).

value them by their size (lines of code). The more a program contains code, the better is its functionality (it has more and better properties), and the more valuable it is from the consumers' point of view. Let the consumers' valuation of the firm's copyright program  $V_R$  be evenly distributed on the interval  $[0, hS]$ , where  $h > 0$  is a parameter transforming program size  $S$  (lines of code) to willingness to pay.

For the open source program, we can distinguish three cases depending the relationship of it and the commercially supplied program. First, the open source program may be a complement to the commercially supplied program. The effects of this complementarity are characterized by a positive externality,  $k > 0$ . It is assumed that due to this complementarity between the programs, consumers' valuation of the commercial software program is increased. The valuation for that product is evenly distributed on the interval  $[0, h(S + ky)]$ . That is, due to the appearance of free open source software, consumers have an access to  $ky$  lines of code, creating valuable additional features.

Secondly and most interestingly, the open source program may be a competing substitute to the commercially produced program, and thus we have a case of negative externality,  $k < 0$ . Notice that consumers' valuation of the firm's copyright program,  $V_R$ , is unchanged and belongs to the interval  $[0, hS]$ . Their valuation of the open source program,  $V_O$ , in turn is evenly distributed on the interval  $[0, -hky]$ . We additionally assume that in the case of substitutes, the ratio of the valuations of the copyright and open source programs is constant for all consumers.

Finally, the open source program may be totally independent i.e. it does not affect the consumer market of commercial software in any way, and quite naturally in this case  $k = 0$ . We notice that this last case of zero externality resembles Spence's original analysis, and indeed in section 3 we demonstrate this to be case in our model.

The proposed structure characterizes an open source programming project in two dimensions. First, the magnitude of parameter  $k$  describes how much of the programming credential transforms into functionality of an open source program that consumers value. Second, the sign of  $k$  expresses the nature of the open source program i.e. whether it competes with or complements the commercial program.

We have defined consumers' valuations and can turn into describing a software company's profit maximizing behavior; it sets the price in order to maximize revenues. As the willingness to pay is uniformly distributed, we have the case of linear demands. When programs are complements, the inverted demand function for the commercial program reads

$$p = h(S + ky) - \frac{h(S + ky)}{M}x, \quad (1)$$

where  $x$  is the number of consumers who will buy the commercial software.

In the case of substitutes it is the marginal consumer that is indifferent between the commercial program and the open source program that determines the demand for the commercial program. Note that the open source program has zero price and we defined  $k < 0$  in the case of substitutes. For the marginal consumer  $i$  with valuations  $V_{Ri}, V_{Oi}$  holds

$$V_{Ri} - p = V_{Oi}. \quad (2)$$

We assume throughout the analysis that the ratio of valuations is constant to all consumers, implying  $\frac{V_{Ri}}{V_{Oi}} = \frac{hS}{-hky}$ . Inserting this in (2) and manipulating yields

$$V_{Ri} = \frac{S}{S + ky}p. \quad (3)$$

Our assumption that willingness to pay for the firm's copyright program is evenly distributed implies that the number of consumers that have a higher willingness to pay than the marginal consumer  $i$  is  $x = \frac{S - V_{Ri}}{S}M$ . Solving for  $V_{Ri}$  and inserting it in (3) yields the inverted demand function for the commercial program in the case of substitutes

$$p = h(S + ky) - \frac{h(S + ky)}{M}x.$$

And as we can observe, the inverted demand functions for complement and substitute commercial programs are identical. The firm's revenue maximization yields the optimum price  $p^* = \frac{h(S + ky)}{2}$ , and then the optimum quantity  $x^* = \frac{1}{2}M$ . The revenue function is both in the case of a complement and a substitute

$$R = \frac{h(S + ky)}{4}M. \quad (4)$$

In fact we can use (4) to represent all cases. In the case of an independent open source program, we defined  $k = 0$ . In such a case, the firm sets an

unconstrained monopoly price and the revenue function is  $R = \frac{hS}{4}M$ . Note that though the revenue functions are identical in the cases of a complement and a substitute open source program, the market outcomes are quite different. For a complement, the same consumers that would have bought the firm's program anyway are willing to pay more of the very program. In the case of substitutes some consumers buy the commercial program and the rest of the consumers acquire the open source program for free. This distinction will become important in welfare analysis. In order to simplify our notation in the rest of the paper we define variable  $a = \frac{hM}{4}$  that captures the "market size effect". The firm's revenue thus becomes

$$R = a(S + ky).$$

In order to be viable the firm has to hire a programmer, and following Spence we assume that the programmers have all the bargaining power in the labor market. This implies that the commercial software companies will compete for the programmers and end up with zero profits

$$\pi = R - w = a(S + ky) - w = 0.$$

## 2.4 The Strategies, solution concept and timing

In the labor market, strategies  $(\bar{y}_B, \bar{y}_G, w^*)$  and a system of beliefs form an equilibrium. In particular, we use the Perfect Bayesian Equilibrium (PBE) as a solution concept. The building block for using it is the assumption that the success achieved within an open source programming project (i.e. received *programming credential*) is regarded by commercial software companies as a credible signal of an individual programmer's innate ability or simply productivity.

We assume that each programmer chooses the amount of programming work to do within the open source project to get a position in a merit-based ranking list given the wage function  $w^*(S, y) = a(S + ky)$  (We remind that  $S = 2$  for a good and  $S = 1$  for a bad programmer). The low productivity "bad type" faces a problem

$$\bar{y}_B \in \arg \max_y [w^*(1, y) - y],$$

and the high productivity "good type"

$$\bar{y}_G \in \arg \max_y [w^*(2, y) - \frac{y}{2}].$$

The software company hires a programmer with a programming credential  $y$  (i.e. position at the merit based ranking list) at wage

$$w^*(S, y) = \beta^*(1 | y)a(1 + ky_B) + (1 - \beta^*(1 | y))a(2 + ky_G)$$

with beliefs  $\beta^*$  that are consistent with strategies. In particular, if the optimal credential levels differ,  $\bar{y}_B \neq \bar{y}_G$ , then if  $y \leq \bar{y}_B$ ,  $\beta^*(1 | y) = 1$  and if one observes  $y \geq \bar{y}_G$ ,  $\beta^*(1 | y) = 0$ . Of course, in the case when the optimal credential levels coincide,  $\bar{y}_B = \bar{y}_G$ , then if observed credential  $y = \bar{y}_B = \bar{y}_G$ ,  $\beta^*(1 | y) = 1 - q_1$ . It is widely known that the predictive power of PBE is weak in a sense that it does not restrain the out of equilibrium beliefs. Thus in the rest of the paper we use the Cho-Kreps intuitive criterion to restrain out of equilibrium beliefs and focus on the least cost separable (lcs) equilibrium, where the low productivity programmer chooses zero credential and the high productivity programmer in turn picks up the smallest possible level of credential that allows separation.

In the product market, the firm sets its price to maximize profits and consumers either buy one unit of commercial copy right program or they prefer one unit of a free open source product.

Timing of the model can be summarized as below:

0. Nature assigns productivities 1, 2 and the proportion of high productivity programmers is  $q_1$ .

1. A programmer may engage himself in an open source programming project thus obtaining a programming credential  $y$  (a status on a merit-based ranking list) and thus creating an OS program with  $|k|y$  lines of program code.

2. A commercial software company is willing to hire a programmer with conditions  $\{w, y\}$ .

3. A company employs a programmer whose status is at least equal to  $y$  at wage  $w$ .

4. A company develops a copyright program of quality  $S$ .

5. Consumers buy the copyright program or acquire the open source program for free.

6. Profits are realized and wages are paid.

### 3 The Analysis

In this section we analyze the optimal behavior of programmers in labor market and demonstrate what are the implications of skill signalling externalities. When individual programmers decide in what extend they participate in OS projects in order to signal their skills they do this by anticipating the wage offer of the commercial software company. In Spence the firm offers a wage that is equal to the expected productivity of employees and all the bargaining power lies with the employee and the firm profit depends only the employee's productivity. In addition to this relationship we present an another one arising from the impact of the open source program in the product market.

To start with it is useful to recall that due to competition in labor market the firm pays its full revenue as wage,  $w(S, y) = a(S + ky)$ . Individual programmers engaging themselves in OS projects choose the optimal credential levels  $\bar{y}_G$  and  $\bar{y}_B$  by maximizing their utility

$$\max_{y_G} U_G = w^*(2, y) - \frac{y_G}{2},$$

$$\max_{y_B} U_B = w^*(1, y) - y_B$$

given the wage function ,

$$w^*(S, y) = \beta^*(1 | y)a(1 + ky_B) + (1 - \beta^*(1 | y))a(2 + ky_G).$$

The incentive compatibility constraints for good and bad types read

$$a(2 + ky_G) - \frac{y_G}{2} \geq a(1 + ky_B) - \frac{y_B}{2}$$

$$a(1 + ky_B) - y_B \geq a(2 + ky_G) - y_G \tag{5}$$

As we are focusing on the *least cost separating equilibrium*, it is clear that  $\bar{y}_B \neq \bar{y}_G$ . Moreover, since getting a credential is costly, it is optimal for the bad type not to get one, i.e.  $\bar{y}_B = 0$ . Therefore, from the binding low productivity type's IC -constraint (5) we can solve  $\bar{y}_G = \frac{a}{1-ak}$ . Thus we have  $\bar{y}_B = 0$ ,  $\beta^*(1 | 0) = 1$  and  $\bar{y}_G = \frac{a}{1-ak}$ ,  $\beta^*(1 | \frac{a}{1-ak}) = 0$ . And the wages are  $w_B^* = a$ ,  $w_G^* = a \left(2 + k\frac{a}{1-ak}\right)$ .

The resulting lcs equilibrium has the following properties. The bad type finds it optimal not to take part in the open source project, and thus

$$\bar{y}_B = 0.$$

The programming credential requires such a large programming undertaking from him that the high wage cannot compensate the disutility of programming work (signalling). The good programmer contributes to the OS project in such a way that that the least-cost credential for the good programmer to separate is

$$\bar{y}_G = \frac{a}{1 - ak}.$$

A possible interpretation of the least cost equilibrium is that the good programmer chooses from numerous potential OS projects the one which just takes him to the merit-based list with activity  $\bar{y}_G$ .

In lcs equilibrium, the utilities of the programmers are

$$\bar{U}_B = a$$

$$\bar{U}_G = 2a + \frac{(2ak - 1)a}{2(1 - ak)}$$

Before stating the actual results in the case when externalities are present, we observe the following. First, we have already mentioned that in the case of no externality, our results resemble those of Spence. To see this let us assume for a moment that  $k = 0$  and in addition that the market size effect,  $a = 1$  as in Spence. In this case we have  $\bar{y}_B = 0$ ,  $\bar{y}_G = 1$ ,  $\bar{U}_B = 1$  and  $\bar{U}_G = 2$  and we have

**Lemma 1** *In the case of zero market externality and in the absence of market size effect, our results coincide with those of Spence (1973, 1974):  $\bar{y}_B = 0$ ,  $\bar{y}_G = 1$ ,  $\bar{U}_B = 1$  and  $\bar{U}_G = 2$ .*

Secondly, it is useful to notice for later purposes that the amount of programming credential the good type has to put forward in order to signal credibly his type depends on the magnitude of market externality, and this of course affects also the utility level of the good type:

$$\frac{\partial \bar{y}_G}{\partial k} > 0, \frac{\partial \bar{U}_G}{\partial k} > 0. \quad (6)$$

As we have now derived the least cost programming credential needed to separate a good type from the bad one we can state some interesting results concerning credential with regard to the market externality:

**Proposition 2** *The least-cost programming credential that separates the good programmer from the bad is higher when the programs are complements than when they are substitutes.*

**Proof.** It is enough to notice that in the case of a complement ( $k > 0$ ),  $\bar{y}_G = \frac{a}{1-ak}$  is larger than in the case of a substitute ( $k < 0$ ). ■

When the open source programming project creates a substitute to the firm's program in the consumer market the good programmer suffers from the market externality. The competition in the consumer market lowers the firm's profit and thus ultimately the wage of the good programmer once he is hired by the commercial software company.

**Proposition 3** *The high productivity ("good") programmer's utility is higher when the programs are complements than when they are substitutes.*

**Proof.** This is clear from  $\bar{U}_G = 2a + \frac{(2ak-1)a}{2(1-ak)}$ . ■

When the OS program is a complement to the firm's commercial program, the increase in disutility resulting from the higher programming credential is dominated by the increased profits and thus wages. The implication of this is that if the good programmers can determine the nature of the OS project, they prefer a project that results in a complement program,  $k > 0$ , and one where as much as possible of the effort is directed to programming activity that consumers ultimately value (large  $|k|$ ).

**Proposition 4** *In the case of a complement OS program and given that the market externality is strong enough,  $k > \frac{1}{a}$ , the good types cannot separate from the bad ones and thus we have a pooling equilibrium of  $\bar{y}_B = \bar{y}_G = 0$ .*

**Proof.** This is clear from  $\bar{y}_G = \frac{a}{1-ak}$  that represents the minimum of programming credential, since in the case of strong externality,  $k > \frac{1}{a}$ ,  $\bar{y}_G$  becomes negative, which of course is impossible. Thus we have a case where both types do not acquire any programming credential at all,  $\bar{y}_B = \bar{y}_G = 0$ . ■

The nature of the OS project matters a lot. From the good programmer's point of view, the most preferred project entails creating a complement to the firm's program. However, the outcome must not be excessively valued by consumers. If  $k > \frac{1}{a}$ , the optimal least cost programming credential to separate does not exist. Why is this so? To state it a bit differently than in the proof above, the incentive compatibility constraint of the low productivity programmer to mimic the good one does not hold for any education credential. The reason is that the market externality effect of the OS work, performed even by the low productivity programmer, dominates the disutility of it.

The firm itself does not care of the nature of the OS project, since it loses all market revenue in the bargaining with the programmer. However, we can raise some general issues based on the results. First of all, if the OS program is a substitute and highly valued by consumers, the good programmer achieves the least cost programming credential with small effort. Is this a dependable signal of productivity in the firm's large project? Of the complement program project, we can ask whether programming a complement, maybe using techniques different from the firm's project, acts as a dependable signal?

#### **4 Do software companies have incentives to support open source projects?**

In the following, we make an assumption that there exists an outside employment option with wage  $w_o$ , and this outside option is accessible for both the low and high productivity programmers<sup>10</sup>. In particular, we want to examine now a situation where the level of outside wage is such that the software company is not profitable if it hires a programmer blindly. That is

$$R = a(1 + q_1) - w_o < 0.$$

Now it is clear that simply to exist in the first place, the company has to screen programmers and then by hiring the high productivity programmer it makes a positive market revenue. As before, since the programmer holds the bargaining power in the labor market, he receives all the revenue as wages. In particular, we want to demonstrate that under these conditions it is in the commercial software company's interest to create and support

---

<sup>10</sup>The analysis has similarities with the textbook model of Hirschleifer and Riley (1992).

open source programming projects and use them as screening devices. One could even interpret this as an outsourcing of personnel management (or recruiting) activities.

For simplicity, we assume that the support for open source projects is materialized in the form of a lump-sum payment  $F$ . Then the profit of the commercial software company reads as  $\pi = a(2 + ky_G^F) - F - w = 0$ , where  $y_G^F$  is determined by the bad programmer's incentive compatibility constraint with respect to the outside wage  $w_o$ ,

$$a(2 + ky_G^F) - F - y_G^F \geq w_o. \quad (7)$$

The least-cost separating level of programming credential is

$$\bar{y}_G^F = \frac{2a - w_o - F}{1 - ak} \quad (8)$$

From (8) we can see that the software company can provide financial support for the OS project up to the difference between the market revenue from its program created by a good programmer and the outside wage,  $F < 2a - w_o$ , and quite naturally the least cost separating credential is decreasing in  $F$ . The consideration we expressed earlier is relevant here also. If the level of credential is very low, it may not be a dependable signal. So we can summarize the above discussion as:

**Lemma 5** *The software companies do have incentives to support open source software projects in order to use them as screening devices for their new labor force.*

## 5 Welfare Analysis

Consider now the possibility that a society as such is able to choose the nature of an open source project, i.e. whether it is a complement or a substitute to commercial copyright program. As we noted the market outcomes are different and this has a bearing on the resulting levels of welfare. We measure welfare as the net of surplus, that is the sum of firm's profits and consumer surplus minus the disutility of the programmer. Again we assume the existence of the outside wage  $w_o$ . Now to get a bit more general view of welfare issues we characterize programmers with productivities  $\theta_G > \theta_B > 0$  instead of 2 and 1. This is simply due to the fact that in the welfare analysis the productivity difference does matter. Now equation (7) in the case when  $F = 0$  yields

$$a(\theta_G + ky_G) - \frac{y_G}{\theta_L} \geq w_o.$$

This in turn implies the that the least cost programming credential to separate is

$$\bar{y}_G = \frac{\theta_B(a\theta_G - w_o)}{1 - ak\theta_B}.$$

In the case of a complement OS program, welfare reads

$$W_C(a, k) = \frac{3}{2}a \left( \theta_G + k \frac{\theta_B(a\theta_G - w_o)}{1 - ak\theta_B} \right) - \frac{\theta_B(a\theta_G - w_o)}{\theta_G(1 - ak\theta_B)}.$$

When program are substitutes, welfare is

$$W_S(a, k) = \frac{3}{2}a \left( \theta_G - \frac{1}{3}k \frac{\theta_B(a\theta_G - w_o)}{1 - ak\theta_B} \right) - \frac{\theta_B(a\theta_G - w_o)}{\theta_G(1 - ak\theta_B)}.$$

The welfare measures allow us to assess whether the result of OS project with a given level of market externality should be either a complement or a substitute to the firm's commercial program from the society's point of view. In order to facilitate comparisons, we use the same magnitude of the market externality effect,  $m$ , in both cases. Figure 1 below depicts how the least cost credentials are determined in both cases. Straightforward comparison of welfare functions gives us

**Proposition 6** *Welfare is higher with a substitute OS program than with a complement when  $a < \frac{2\theta_B - \theta_G}{2\theta_G\theta_B m}$ , where  $m = |k|$  is the level of market externality of the open source program.*

**Proof.** *Developing the inequality  $W_C(a, m) < W_S(a, -m)$  yields the following inequality.*

$$\frac{3}{2}a\theta_G + \frac{3am}{2(1 - am\theta_B)} - \frac{1}{\theta_G(1 - am\theta_B)} < \frac{3}{2}a\theta_G + \frac{am}{2(1 + am\theta_B)} - \frac{1}{\theta_G(1 + am\theta_B)}$$

*It simplifies into*

$$a < \frac{2\theta_B - \theta_G}{2\theta_G\theta_B m}.$$

■

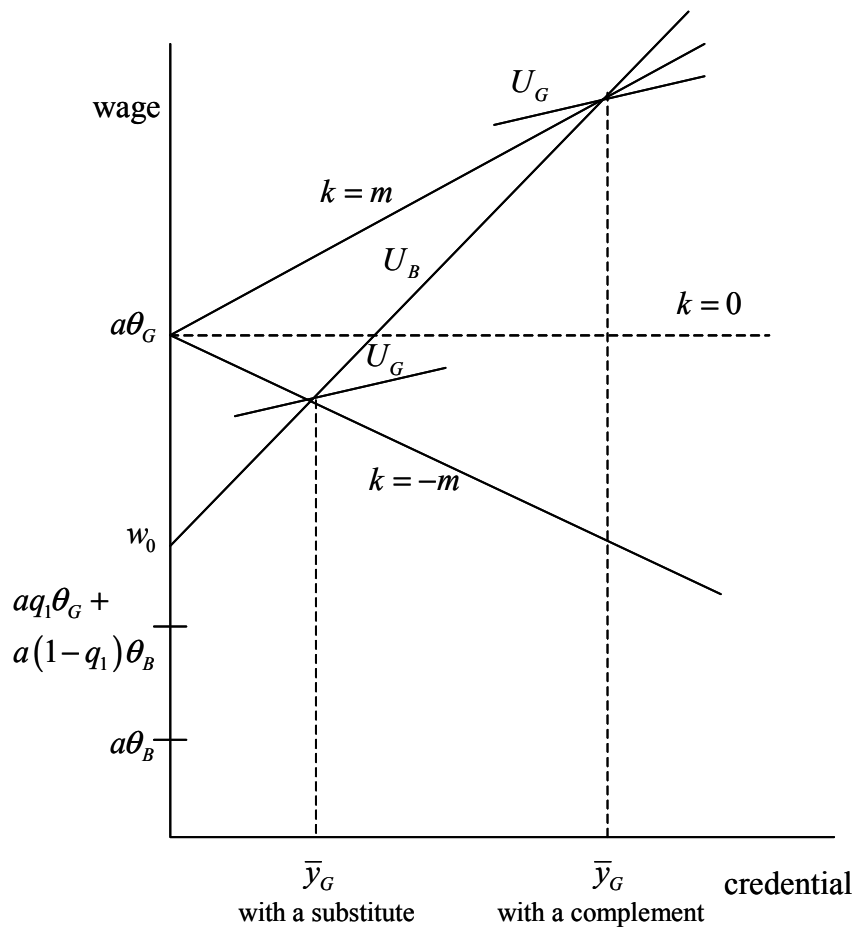


Figure 1: The optimal programming credential in the case of a substitute and a complement OS program

Recalling that  $a = \frac{hM}{4}$ , the condition implies that when the consumer market is small compared to the programmer population or when consumers have low valuations of the programs, the welfare associated with OS project resulting in a substitute program is higher. With our specifications, larger welfare with a substitute open source program is possible if the productivity difference between the programmers is small; the good programmer is less than twice as productive as the bad one. The threshold market size is decreasing in the market externality of the OS project,  $m$ . When market size increases, profits and consumers' surplus dominate the good programmer's increased disutility of separation with a complement open source project.

Note also that the market outcomes differ: with a substitute, all consumers use some program, whereas with a complement only a half of consumers use the firm's program and the complementing OS program. If distributional considerations are included in the analysis, the threshold of the proposition is likely to be higher.

We proved earlier that for a given level of market externality, the good programmer always prefers an open source project that results in a complement program. The welfare analysis shows that when the condition of the proposition holds, society would choose a project producing a substitute, since the programmer does not internalize the beneficial effects coming via the consumer's surplus.

## 6 Conclusion

This paper has extended the classic article of job market signalling by Spence into the situation where the signalling activity itself may have interesting externality effects that are coming through via products markets. In particular, we considered two cases where the product developed within the open source project and the commercial program were either complements or substitutes. As was also demonstrated, the case of no product market externality resembles the classic result of Spence. The particular example that we chose to emphasize was related to job signalling in the software industry. However we believe that the phenomenon exists in other areas too. Consider, for example, that a Master's degree in a university requires the completion of a thesis. This literary output might compete with or complement a commercial (text)book of the potential employer interested in hiring the graduate. Our focus on OSS is motivated by the 'virtuality' of software and the resulting possibly strong market responses to the availability of free OS software. Of course, it is important to realize that our analysis is partial

in a sense that we focus on one market niche only. In reality, it may very well be that even if the OS software is independent in this particular market niche we examine, it may well have external (positive or negative ) effects on some other market niche. This is a natural question to be examined in future studies.

## References

- [1] Bessen J., 2001, Open source software: Free provision of complex public goods, mimeo, [www.researchoninnovation.org/opensrc.pdf](http://www.researchoninnovation.org/opensrc.pdf), (Accessed April 2002), Research on innovation.
- [2] Browne C., 1999, The economics of free software, [www.ntlug.org/~cbbrowne/freecon.html](http://www.ntlug.org/~cbbrowne/freecon.html) (accessed January 2000).
- [3] Collab.net (2001) Open source networks, [www.collab.net](http://www.collab.net) (accessed October 2001).
- [4] Dasgupta P. and P. David , 1987, Information disclosure and the economics of science and technology, CEPR Discussion Paper 73, also in: Feiwel D ed., Arrow and the ascent of modern economic theory (New York University Press, New York).
- [5] Dasgupta P. and P. David, 1994, Toward a new economics in science, *Research Policy* 23(5), 487-521.
- [6] Dewatripont M., I. Jewitt and J. Tirole, 1999, The economics of career concerns I&II, *Review of Economic Studies* 66(1), 183-217.
- [7] GNU Project, 2000a, Categories of free and non-free software, [www.gnu.ai.mit.edu/philosophy/categories.html](http://www.gnu.ai.mit.edu/philosophy/categories.html) (accessed June 2000).
- [8] GNU Project, 2000b, What is copyleft? [www.gnu.ai.mit.edu/copyleft/copyleft.html](http://www.gnu.ai.mit.edu/copyleft/copyleft.html) (accessed June 2000).
- [9] Hamm I., Roberts J., Slaughter S. and Fielding R., 2002, Delayed Returns to Open Source Participation: An Empirical Analysis of the Apache HTTP Server Project, Graduate School of Industrial Organization, Carnegie Mellon University, <http://www.idei.asso.fr/English/EPresent/index.html> (Accessed September 2002).

- [10] Hirschleifer J. and J. Riley, 1992, *The analytics of uncertainty and information*, Cambridge University Press, Cambridge.
- [11] IBM News, 2001, [www.ibm.com/news](http://www.ibm.com/news) (Accessed December 2001).
- [12] Johnson J., 2002, Open source software: Private provision of a public good, *Journal of Economics and Management Strategy*, 11(4), 637-662.
- [13] Koboldt C., 1995, Intellectual property and optimal copyright protection, *Journal of Cultural Economics* 19(2), 131-155.
- [14] Kogut B. and A. Metiu, 2001, Open source software development and distributed innovation, *Oxford Review of Economic Policy* 17(2), 248-264.
- [15] Lerner J. and J. Tirole, 2002, Some simple economics of open software, *Journal of Industrial Economics* 50(2), 197-234.
- [16] Linux-PAM, 2003, <http://www.kernel.org/pub/linux/libs/pam/HallOfFame.html> (Accessed January 2003).
- [17] Mustonen M., 2002a, Copyleft - the economics of Linux and other open source software. *Forthcoming in Information Economics and Policy*.
- [18] Mustonen M., 2002b, Why do firms support the development of substitute copyleft programs?, Discussion paper 529, Department of Economics, University of Helsinki.
- [19] Netcraft, 2001, *The netcraft web server survey*, Netcraft, Bath, England.
- [20] Raymond E., 1998, *The cathedral and the bazaar*, [www.firstmonday.dk/issues/issue3\\_3/raymond/index.html](http://www.firstmonday.dk/issues/issue3_3/raymond/index.html) (Accessed April 2002).
- [21] Schiff A., 2002 *The economics of open source software: A survey of the early literature*, *The Review of Network Economics* 1(1), 66-74.
- [22] Spence, M. 1973: Job Market signalling, *Quarterly Journal of Economics* 87. 355-74.
- [23] Spence A.M., 1974, *Market signaling*, Harvard University press, Cambridge, USA.
- [24] Stern S., 1999, Do scientists pay to be scientists? *NBER Working Paper* 7410.