Markku Stenborg[1]

Waiting for F/OSS: Coordinating the Production of Free/Open Source Software

This version August 7, 2004

[1] Contact information: Markku Stenborg, ETLA (The Research Institute of the Finnish Economy), Lönnrotinkatu 4 B, FIN-00120 Helsinki, Finland, markku.stenborg@etla.fi. This article is a part of the joint research program of BRIE, the Berkeley Roundtable on the International Economy at the University of California at Berkeley, and ETLA, the Research Institute of the Finnish Economy (brie-etla.org). Financial support from Nokia and the National Technology Agency (Tekes) is gratefully acknowledged. All opinions expressed and errors made are those of the author.

Markku Stenborg, Waiting for F/OSS: Coordinating the Production of Free/Open Source Software.

Abstract

Costs of waiting for Free/Open Source Software (F/OSS) program to be released are borne by all agents until enough modules have been produced. Trade-off between producing a F/OSS module and free-riding is analyzed as a game of war of attrition, in which modules to be developed and potential developers are heterogeneous. It may be optimal to volunteer for high-profile module that creates reputation and signals programming ability. It may be optimal to volunteer strategically for low-profile module even if high-profile module is available to speed up development process and reduce costs of waiting. Even if waiting brings the opportunity to free ride, there may be a rush to develop high-profile module at the first opportunity. The model provides an explanation for how large-scale F/OSS projects can be coordinated without markets and prices nor hierarchies such as firms.

Keywords: Open Source Software, War of Attrition, Coordination

Running head: Coordinating F/OSS Production

1 Introduction

A user-programmer can either buy a proprietary product that solves her needs, do the programming herself or wait for somebody else to produce an appropriate piece of software and to publish it on the Internet. Until enough modules have been published, a large-scale Free or Open Source Software (F/OSS) project, such as Linux, is essentially useless, and all users must bear the opportunity costs of waiting for the completed software package. This paper studies the coordination problem of the production of F/OSS modules as a waiting game.

There are two key economic questions in F/OSS development: the motivation of individuals and the coordination of effort. First, there is a growing body of economic literature that attempts to explain the private incentives to provide F/OSS, and the topic is briefly discussed in Section 2 below. Second, note that some F/OSS are large and highly complex products. For instance, the development of Red Hat Linux 7.1 distribution is estimated to have required about 8,000 person-years of development time, with a cost over $ 1 billion, if produced by conventional proprietary means in the U.S. (Wheeler, 2002). These complex projects are somehow coordinated without the help of neither markets and prices nor hierarchical control such as in firms. To my knowledge, this is the first paper to look at the economics of coordination of F/OSS development. Further, most existing economic papers look at the incentives to participate in F/OSS development in static setting, in which agents make only one decision regarding whether to participate on a F/OSS project or not. My paper adds a time dimension to this problem.

Many programming tasks are most efficiently performed by a single individual. This leads to a waiting game: the longer one waits, the more likely it is that someone else will provide a F/OSS module. But until somebody has developed that module, everyone has to wait, which is a cost to all. Bliss and Nalebuff (1984) were the first analyze this type of a question using a war of attrition. Bliss and Nalebuff show how time plays the role of a screening device in the presence of private information about the costs of providing the service. Bilodeau and Slivinski (1996) analyze a complete information and finite horizon version of that waiting game, and show that, ceteris paribus, the individual for whom the benefit/cost ratio of performing the public service is largest, the most impatient to consume it, or the one who stands to benefit from it the longest is most likely to provide the service. Bitzer and Schröder (2003) use Bilodeau and Slivinski's model to analyze production of a single type of F/OSS in the setting of complete information and finite horizon.

However, software modules are heterogeneous – effort and other private costs required as well as reputation, signaling and other private benefits earned by the developer vary between modules, and these are private information. Being credited for major developments of Linux kernel might generate higher hacking reputation and provide clearer signal to labor or capital markets about the developer's talent than being credited for the development of device drivers, say. Also the potential developers are heterogeneous, as they have varying preferences for hacking reputation, for labor or capital market signals and different opportunity costs.

Bulow and Klemperer (1999) analyze a generalized war of attrition, in which $N+K$ players are competing for N prizes, and where each player has private information. That model can be modified to analyze the coordination of the production of heterogenous F/OSS modules under incomplete information. I provide an explanation for coordination of programming effort without market prices or formal hierarchical control such as in firms. Here, "time is money": time and opportunity costs of waiting guide the decisions. The individuals who gain most from the F/OSS relative to their programming costs are most eager to provide the modules, and the individuals who have high programming costs relative to private and public benefits from software are most willing to wait the longest for somebody else to publish F/OSS modules. My paper owes most to Bilodeau and Slivinski (1996), Bitzer and Schröder (2003), Bulow and Klemperer (1999) and, especially, Sahuguet (2003), who modifies Bulow and Klemperer's model to public goods.

The paper is organized as follows. In the next Section, I shall briefly discuss the economics of F/OSS. Section 3 presents the formal model, and Section 4 looks at the full information case. In Section 5, I start the analysis of the incomplete information version with the second-stage subgame. Sections 6 and 7 analyze separating equilibrium and Section 8 pooling equilibrium. Section 9 concludes. The proofs are found in the Appendix.

2 Some Economics of Free/Open Source Software

F/OSS is privately produced public good. According to the established economic theory, private provision of a public good is Pareto inefficient, as individuals face an incentive to free ride on the contributions of others. Second, many programming tasks are often best performed by a single individual, who bears the entire cost of providing a software module which benefits the whole community. Individuals then have strong incentives to free ride. Third, a key element here is voluntary selection of tasks, as each programmer is free to choose on what she wishes to work on. Large scale F/OSS such as Linux are subject to collective provision constraint, as their production depends on coordinated contri-

butions from a large number of developers. Smith and Kollock (1999, p. 230) have even called Linux "the impossible public good". Thus, large scale F/OSS ought to be at the worse end of the spectrum of public goods.

Contrast the theory with the facts. Currently, there exists thousands of F/OSS projects, ranging from small utilities and device drivers to large and complex packages such as Linux, Apache, Open Office and MySQL. F/OSS has proved to be successful mode of innovation and software production.[1] For instance, GNU/Linux and other free UNIX-based operating systems are the only real challengers to the Microsoft Windows operating system for Intel-based PCs, Sendmail routes at least 42% of mails in the Internet,[2] and Apache dominates the web server market.[3] Arguably, some of the F/OSS products are of better quality than competing commercial products. For instance, Kuan (2001) tests the quality of software using bug resolution rates as a proxy for quality, and finds support for the hypothesis that F/OSS outperforms commercial proprietary software.

Some explanations for this paradox have already been proposed. Raymond (1999) offers ego gratification as an explanation, based on observations that hackers are motivationally like artists, in the sense that they seek fun, challenge, and beauty in their work. Private benefits as an idea to explain F/OSS production can also arise through reputation and signaling, through performance comparisons and reputations for expertise. Johnson (2002), Lee et. al. (2003), Leppämäki and Mustonen (2003), Lerner and Tirole (2002) and Mustonen (2003) claim that programmers behave as to be appreciated by their fellows and like to show off their abilities.[4] The economics approach adds delayed monetary compensations to that story. Peer recognition creates a reputation that can be monetized in the form of job offers, privileged access to venture capital, etc. Also my paper provides an explanation to the F/OSS paradox that is partly based on these ego gratification, reputation and signaling arguments. My paper adds the timing element in the signaling and reputation stories, as a player can wait for another volunteer to produce the piece of F/OSS before her.

---

1 For more on F/OSS and related political economics, see, e.g., Weber (2004).

2 http://www.dwheeler.com/oss_fs_why.html

3 According to the Netcraft survey, http://www.netcraft.co.uk/Survey/, in January 2003, over 65% of all active web sites use Apache.

4 In addition, Mustonen (2002) offers an explanation based on increased compatibility between proprietary and free programs that increases the value of the proprietary program, providing incentives to support F/OSS efforts. Bessen (2002), Johnson (2002) and Kuan (2002) note that individual user-programmers know their own preferences better than a firm does and that a greater skill set can be exploited. Bessen's explanation hinges on the complexity of the software and Kuan models open source as customer vertical integration into production. See also Weber (2004), Chapter 5.

Reputation and signaling can arise as F/OSS is organized such that every significant contribution can be traced back to the original author. For instance, in one of the biggest F/OSS-projects, the Linux kernel, there exists a public "changelog" file which lists all those programmers who have contributed to the official source and their specific inputs. Each proposal to modify the code undergoes a peer review process and only those modifications sanctioned by the referees make their creators legitimate authors.[5] A spot in the credits thus serves as a valuable signal on job and capital markets characterized by asymmetric information. The ex ante expected value of the deferred payoff makes striving for the signal worthwhile since the unrestricted access to the Linux kernel code and its changelog file allows for the right interpretation and honoring even by outsiders ex post. Contributions to F/OSS are not only unselfish donations or the pursuit of ego gratification, but also investments based, e.g., on future career concerns.

3 Model

Consider a group of N risk-neutral individuals, each of whom has the ability to develop one discrete software module for the F/OSS project consisting of two software modules.[6] Until the entire F/OSS project is finished, all agents need to pay an opportunity cost of waiting, as they need to use an alternative commercial closed-source program, suffer from bugs or from dysfunctional Application Program(ming) Interfaces, or in some other manner find the existing software less than adequate. Following Bulow and Klemperer (1999), Sahuguet (2003) and others, I normalize the cost of waiting equal to 1 per unit time until the entire software project is completed. There is no discounting.[7]

Once a programmer decides to develop a F/OSS module, she pays a one-time private cost C, discussed below, and publishes the software for free. The public benefits from the project are delayed until the whole project is complete. The first programmer then pays a waiting cost of c per unit time, $0 \leq c \leq 1$, until the project is completed. For $c < 1$, the

---

[5] For more on this issue, see Weber (2004), especially Chapter 4.

[6] In the formal model, I will assume $N = 3$. Bulow and Klemperer's model of generalized war of attrition gives us tools to analyze N individuals developing K modules but here a model with more than three players or with more than two modules would be more cumbersome without adding any major insights to the strategic trade-offs (see also Sahuguet, 2003, footnote 3).

[7] This is without loss of generality. Discounting is equivalent to an exogenous flow of probability that the game ends and agents will stop accruing benefits and costs. Discounting leaves the formulae for optimal stopping time unchanged, but increases the costs of waiting relative to the discounted value of free riding. See also Bulow and Klemperer, p. .

first programmer is able to draw some private benefits, such as reputation or signaling, from her efforts, even if the F/OSS project is still incomplete.[8]

Completed software is non-rival and non-exclusive, and brings benefits to all individuals over some period of time. Once the project is completed, all individuals receive the flow utility value from the F/OSS being available. *B(i)* denotes the total present value of public benefits player *i* draws from a completed project.[9]

Programmers are heterogeneous, as each might have different opportunity costs for producing F/OSS, place different value on reputation and signaling, e.g., due to differences in future career plans, or in some other manner value programming chores and the finished modules differently. Assume that the differences between the individuals can be captured by a one dimensional private type θ. Types are drawn independently from a distribution $F(\theta)$, with $F(\underline{\theta}) = 0$ and $F(\bar{\theta}) = 1,$ that has a derivative f which is strictly positive and finite on $\left[\underline{\theta}, \bar{\theta}\right]$. Write $h(\theta)$ for the hazard rate $\frac{f(\theta)}{1 - F(\theta)}$. Agent's type θ as well as the value of benefits $B(\theta)$ and the net development costs $C_j(\theta)$ are private information to that agent.

Programming requires time, effort, etc., that is costly to the agent. The net cost of developing software module *j* consists of a one-off actual development cost and a net utility flow incurred during some period of time after the module has been released. The utility flow is here due to the value of improved reputation within the community of programmers, the value from signaling of programming ability to labor or capital markets, or any other private benefits above $B(\theta)$ the programmer with type θ is able to draw from working on software module *j*. The net costs of developing the software module *j* at time *t* are then measured by the present value of the net costs, and denoted by $C_j(\theta)$.[10]

The modules to be developed are also heterogenous. There is a "high-profile" module that has lower private opportunity costs than a "low-profile" module. The former requires less effort, involves interesting rather than mundane programming tasks, has higher

---

[8] The analysis of a more general case where there are some public benefits from a partially complete F/OSS project is left for the future work.

[9] Also *B()* is defined with respect to the waiting costs. For instance, an individual who bears no opportunity cost of waiting for the F/OSS has $B() \leq 0$.

[10] The programming cost *C* is also defined with respect to the fixed waiting cost of 1 per unit time. For instance, an individual who bears no opportunity cost related to waiting, e.g., as she finds the existing software more than adequate for her purposes and has no use for hacking reputation, has an infinite net programming cost.

reputation and signaling value, or is in some other manner preferred by the programmers, ceteris paribus. The expected total net costs for an agent with type θ of exerting effort for the high-profile and the low-profile task are $C_H(\theta)$ and $C_L(\theta)$, respectively, and assume that $C_H(\theta) < C_L(\theta)$ for all θ. The payoffs from producing the low-profile and the high-profile task are then $U_L(\theta) = B(\theta) - C_L(\theta)$ and $U_H(\theta) = B(\theta) - C_H(\theta)$, respectively, and the payoffs from free riding are $U_F(\theta) = B(\theta)$. Assume $U_L(\theta) < U_H(\theta) < U_F(\theta)$ for all θ.

I assume that $C_j(\theta)$ are increasing and $B(\theta)$ are decreasing in θ for all.[11] That is, an individual with a high type has high programming costs or places a large valuation for free riding, e.g., as she faces large learning costs, expects to enjoy only small rents from signaling her programming ability or does not value the reputation to be earned within the hacking community.

Time is continuous, and players can choose to produce a module at any point in time.[12] Players also decide which module to choose, if there are more than one left. Their decisions are allowed to depend on observations, e.g., on past actions. If no one has produced a module at time t, each player draws inferences about the types of the other players. Here, this information process is predictable, as will become clear below.

Let θ_R be the lowest possible remaining type at any point of the game, i.e., the type with the lowest cost that has not yet produced a module, conditional on all other players having followed their equilibrium strategies up to that point in time. Let $T(\theta; \theta_R)$ denote the additional amount of time that the player with type θ who has not yet released a module will wait to do so, unless somebody else releases it prior to her. Let $P(\theta; \theta_R)$ denote the type θ player's probability for outlasting the waiting game and being able to free ride.

Player's decision reveals information about her type. But note that here the only piece of information revealed is the fact that players do not observe an earlier decision to develop a module from another player. Hence, they can predict in advance what information they

---

[11] As I concentrate on symmetric equilibrium, I shall often drop the notation for individual i and identify a player with her type θ.

[12] Notice, that here the development of a module does not take any time, as the decision to start programming and the release of a complete module are simultaneous. We can think that i's decision to start working on a module is announced publicly, the announcement is a credible commitment to complete the project, and that the announcement preempts others from working on this module. The more realistic case where the time between development decision and the release od final module is explicitly included in the model seems only to complicate the analysis without bringing new insights for the problem of coordination of effort in F/OSS development.

will learn. Then a strategy for a player can be summarized by a type-dependent time to wait before programming a module in the three-player stage and the software module she would select, and an additional time to wait in every possible two-player subgame. These subgames are characterized by the remaining programming task and the updated beliefs about the other player's type.

Here, the beliefs consist of updating the probability distribution of the type of the remaining players only. A deviation from the equilibrium strategies can only be observed if a player released a module, so a deviation removes the deviating player out of the waiting game. Hence, beliefs can always be computed according to Bayes' rule, even for histories that are off the equilibrium path. For the players that did not produce F/OSS module, the updating will merely eliminate the types that should have done so before the actual decision occurred.

I shall only look at the symmetric perfect Bayes Nash equilibria. There obviously exist numerous asymmetric equilibria of this game,[13] but as Farrell and Saloner (1988) and Bolton and Farrell (1990) argue, asymmetric equilibria are unconvincing and inappropriate for the study of decentralized coordination mechanisms. For instance, it is not clear how agents would decide between those equilibria without assuming some explicit coordination. Further, games in continuous time face technical difficulties, which I shall completely avoid. I assume that after a player releases a module, a new subgame arises and that strategies are defined with respect to this subgame.

4 Full Information

In this Section, I shall first look the case in which the types and all the costs and benefits are common knowledge, as in Bitzer and Schröder. First, consider a static game. If we model the problem as a simultaneous one-shot game with the strategy set {*produce L*, *produce H*, *free ride*}, the problem becomes a static game of Chicken, in which the winners are able to free ride and receive the payoff $B(\theta)$, and each of the losers develops one module and receives the payoff $B(\theta) - C_j(\theta)$. If no one develops F/OSS, the payoff for all is 0. Given the specifications above and the assumption of full information in static setting, I can state the following.

Lemma 1. Let θ^H be an individual such that $B(\theta^H) - U_j(\theta^H) = 0$. No individual with type $\theta \geq \theta^H$ ever volunteers for the task $j = L,H$ in a static game.

[13] See, e.g., Ponsati and Sákovics (1995).

In other words, only those individuals who face low development costs or who benefit most from the F/OSS, directly or indirectly, are the potential developers of the software. The community of potential developers for module j then consists of individuals with $\theta \leq \theta_j^H$.

This game has a large number of pure and mixed strategy Nash equilibria, in which anyone might be the actual developer of the software. Hence one cannot deduce who will actually develop the F/OSS, nor can we analyze the coordination of who shall produce which module.

We can expand the strategy sets and allow the agents the option of postponing the decision to produce F/OSS so that they can wait and see if someone else develops the modules instead. The length of time an agent is willing to wait naturally depends on the benefits she would gain if the F/OSS were to exist, and on the net costs of developing the module herself. In the normal form version of this game, a pure strategy for a player with type θ consists of a two-tuple of time to wait $[T_1(\theta), T_2(\theta; \theta_R)]$, $T_i(\theta) \in [0,\infty)$, where $T_1(\theta)$ denotes the time when type θ will produce the first module and $T_2(\theta; \theta_R)$ denotes the time when θ will produce the second module when type θ_R volunteered for the first task, unless somebody did so at $T < T_i(\theta)$, and the module $j(\theta) = L,H$ to choose, if more than one module is left.

Consider first the "second-stage" two-player war of attrition subgame for module j starting at the time the first programmer chooses a module other than j. The present value of programming module j at time T_2 is $B(\theta) - C_j(\theta) - T_2$, which is decreasing in T_2.

Lemma 2. Any individual with type $\theta \leq \theta_j^H$ will develop the F/OSS voluntarily and immediately at time $T_2 = 0$ in the second-stage war of attrition associated with the module j.[14]

Lemma 2 states that an individual who gains considerably from reputation or signaling programming ability, simply develops the software at the first opportunity, rather than waits for someone else to provide it. By Lemma 2, the second-stage subgame ends at time $T_2 = 0$. With similar reasoning, the first-stage game ends at $T_1 = 0$, i.e., an agent that benefits enough from the F/OSS module volunteers immediately.

Lemma 3. Any individual with type $\theta \leq \theta_j^H$ will develop the F/OSS voluntarily and immediately at time $T_1 = 0$.

Thus every individual in the community of potential developers would rather develop the F/OSS immediately than live without it forever, so the game ends immediately.

A more complex game emerges under the assumption that $C(\theta) < 0$, i.e., when the private net costs outweigh private net benefits. The game now becomes an N player continuous time war of attrition. Using the existing results in Hendricks et al. (1988) and Bilodeau and Slivinski (1996), one can characterize the following equilibria for this type of game. For every individual i, there is a subgame perfect equilibrium outcome in which only i will develop the OSS immediately. If no one but i develops a module, then i's best strategy is to develop that module immediately, and if i develops the module immediately, best reply for everyone else is to wait. The game then has many subgame perfect equilibria – as is usual for this type of game – in which anyone might volunteer, and it is not possible to characterize the individual who will actually provide the software module. Assuming a finite time horizon Bitzer and Schröder (2003) show that a unique subgame perfect equilibrium exists, in which the individual with the lowest θ volunteers at time $t = 0$. That is, in equilibrium, F/OSS module is produced at the first opportunity.

However, the assumptions that the public value of finished software, the private value drawn form the module programmed, and the net development costs are common knowledge are strong. Below I shall look at the problem when information is incomplete and where there is no arbitrary bound for the potential waiting time.

5 Two-player subgame

Now, turning to the incomplete information version of the game, as in Bulow and Klemperer and Sahuguet, let us first analyze the behavior in the last subgame – that is, the subgame following the first decision to program a F/OSS module. Here, there are two agents, and one is needed to produce the second module to complete the F/OSS package, while the other is able to free ride. This is a standard war of attrition game.

The two-player subgames, beginning after the choice of the first programmer, are characterized by a single value $W(\theta)$, where $W(\theta) = U_H(\theta)$ or $W(\theta) = U_F(\theta)$, depending on the choice of the first volunteer. Let $F(\theta; \theta_R)$ be the distribution of types, where θ_R denotes the lowest possible type that has not yet produced a module, conditional on all other players thus far having followed their equilibrium strategies

---

[14] The proofs are found in the Appendix.

$$F(\theta;\theta_R)=\begin{cases}0 \text{ if } \theta<\theta_R \\ \dfrac{F(\theta)}{1-F(\theta_R)} \text{ otherwise}\end{cases}.$$

Let $T_2(\theta;\theta_R)$ denote the equilibrium stopping time for type θ in the subgame starting at the programming decision by type θ_R, i.e., it is the time at which a player of type θ released a F/OSS module, conditional on beliefs θ_R. $T_2(\theta;\theta_R)$ is scaled such that it measures the waiting time in that subgame only. Recall $P(\theta;\theta_R)$ denotes player θ's probability of being able to free-ride, conditional on type θ_R having produced the other module.

Proposition 1. In any equilibrium, for all θ_R, the equilibrium strategy $T_2(\theta;\theta_R)$ is strictly increasing in θ, and $P(\theta;\theta_R)$ equals the probability that player with type θ is able to free ride, conditional that the other remaining player's value exceeds θ_R,

$$P(\theta;\theta_R)=\frac{F(\theta)-F(\theta_R)}{1-F(\theta_R)}.$$

Proposition 1 basically states that players that have low effort costs or who place high value on benefits such as reputation and signaling develop F/OSS early rather than wait in hope of free riding.

Proposition 2. There is at most one symmetric equilibrium in the subgame.[15]

The reason behind the Proposition 2 is that the difference between the expected surplus of any two types is uniquely determined by the standard incentive compatibility arguments. Since any agent's probability of being able to free-ride is fixed by Proposition 1, the difference between the two types' waiting costs must also be uniquely determined. If there were two different equilibria specifying different waiting times $T_2(\theta;\theta_R)$, these two equilibria would yield different differences between the type's waiting costs for at least one pair of types. Hence, the equilibrium in the subgame must be unique.

Proposition 3. The unique symmetric Bayesian Nash equilibrium of the two-player war of attrition subgame is characterized by a time to wait before conceding $T_2(\theta;\theta_R)$ such that

[15] This is a special case of Lemma 2 in Bulow and Klemperer and Proposition 6 below, so the proof is omitted.

$$T_2(\theta;\theta_R) = \int_{\underline{\theta}}^{\theta} W(t)h(t)dt.$$

Proposition 3 describes the equilibrium behavior: waiting times are increasing in type, as high-cost types are more willing to wait longer. Each player knows that her decision to develop a module is relevant only when the other remaining player has not yet done so. The subgame is strategically equivalent to a static game in which players simultaneously commit to waiting times.[16]

The intuition behind Proposition 3 is the following. At each moment, the marginal individual type θ that the equilibrium calls for to program a module, has to be indifferent between producing one now and waiting infinitesimally more to let types between θ and $\theta+d\theta$ to produce the module. The cost of waiting corresponds to $T'(\theta)\cdot d\theta$, the time needed for types between θ and $\theta+d\theta$ to concede. This must be equal to the benefit of waiting more, that is the value of free riding $W(\theta)$ times the probability $f(\theta)/[1-F(\theta)]\cdot d\theta = h(\theta)\cdot d\theta$ that the other player will develop the module during this time interval.

Proposition 4 characterizes the expected surplus of an agent of type θ.

Proposition 4. The expected surplus $S(\theta)$ of an individual of type θ in the two-player war of attrition subgme is

$$S(\theta) = \int_{\underline{\theta}}^{\theta} W'(t)F(t)dt.$$

Proposition 5 presents the expected length of the second-stage two-player waiting game.

Proposition 5. The expected length T^e of the two-player war of attrition is equal to

$$T^e = \frac{1}{2}E\{\min[W(\theta_1), W(\theta_2)]\} = \int_{\underline{\theta}}^{\bar{\theta}} W(t)f(t)[1-F(t)]dt.$$

Recall $W(\theta) = B(\theta) - C_j(\theta)$. Note that the higher the private cost C_j of the module is, ceteris paribus, the more time it takes to find a programmer, which is intuitive.

---

[16] This proposition can also be found in, e.g., Bliss and Nalebuff (1984) and Bulow and Klemperer (1999).

This subgame can be interpreted as a second-price all-pay auction,[17] in which both players pay the bid of the second highest bidder (bear the cost of waiting). The highest bidder (the one willing to wait the longest) wins the prize (free-rides), and both bidders pay the lowest bid, i.e., the second price (both wait until the lower of the remaining types produces the remaining module). In an equilibrium, the prize goes to the type that has the highest value for it and the surplus of the lowest type is zero. The Revenue Equivalence Theorem[18] applies. The expected cost per player is then exactly the expected length T^e of the war of attrition, which is equal to half the expected price paid by the winner in a second price auction. The price paid by the winner in the second-price auction is the expected value of the smaller of the two bids, which is also equal to the expected value of the minimum of the players' valuation since, in a second price auction, bidders reveal their true valuation.

6 Separating Equilibrium

Now turn to analyze the "first stage" waiting game. Here, there are two factors that enter into the players' decisions. First, there is a trade-off between programming the high-profile module and waiting longer. Waiting brings the opportunity to free-ride, at the risk of being stuck with the need to develop the low-profile module, without which the public benefits $B(\theta)$ of the F/OSS project are not realized. Second, the first programmer must make a choice between the high-profile and the low-profile module. It may seem obvious that, in equilibrium, she should always develop the high-profile module, as the low-profile module has higher private net cost. This need not be the case. In this Section, I shall look at a monotonic symmetric equilibrium in which the high-profile module is chosen first, and derive conditions under which such an equilibrium exists. I shall look at some other cases below in next Sections.

In a separating equilibrium, waiting times are strictly increasing in types, so there is a one-to-one mapping between types and waiting times. Beliefs can then be updated in a simple way: after observing a relese of a module, the remaining players compute the type θ_R that corresponds to this observed waiting time. The updated beliefs about the type of the remaining player is characterized by the posterior distribution $F(\theta; \theta_R)$. Recall, $F(\theta; \theta_R)$ is the truncated distribution at the point of time that corresponds to the type that released a module at that time; that is

---

[17] Bulow and Klemperer (1999), Corollary p. .

[18] See, e.g., Myerson (1981), Riley and Samuelson (1981) or Klemperer (2003).

$$F(\theta;\theta_R)=\begin{cases}0 \text{ if } \theta<\theta_R \\ \dfrac{F(\theta)}{1-F(\theta_R)} \text{ otherwise}\end{cases}.$$

Recall also that $T_1(\theta)$ denotes the waiting time for the first module, and $T_2(\theta;\theta_R)$ denotes the additional equilibrium waiting time of a player of type θ in the "second stage" subgame after type θ_R has released the first module,

$$T_2(\theta;\theta_R)=c\int_{\theta_R}^{\theta}U_F(t)h(t)dt.$$

In an equilibrium, no player can have unilateral incentives to deviate, if all other players follow the equilibrium strategies. The usual incentive compatibility construction that characterizes the equilibrium is here as follows. The expected payoff of a player θ using type θ' strategy, denoted by $U(\theta,\theta')$, when all the other players are following the equilibrium strategies is

$$U(\theta,\theta')=\int_{\underline{\theta}}^{\theta'}2f(t)\left[1-F(t)\right]\left[S(\theta,t)-T_1(t)\right]dt$$
$$+\left[1-F(\theta')\right]^2\left[U_H(\theta)-T_1(\theta)\right]-c\int_{\theta'}^{\bar{\theta}}U_F(\theta)f(t)\left[1-F(t)\right]dt$$

The equation notes the fact that player type θ using type θ' strategy is either the first to develop the high-profile module or another player does so before her. The second and the third term on the right hand side of the equation are the following. With the probability $[1-F(\theta')]^2$, the player with type θ using type θ' strategy is the first to choose the high-profile module. After waiting $T_1(\theta')$, she gets $U_H(\theta)$. Then she has to wait until the second module finds a programmer, and must pay a cost c per unit time, $0 \le c \le 1$, for the expected length of the "second-stage" war of attrition. The first term on the right hand side of the equation tells us what happens when another player is the first one to develop at time $T_1(\theta')$. In this case, that player gets the expected surplus $S(\theta,\theta')$ of a type θ in the subgame starting when a type θ' player develops the high-profile module. This case occurs with the probability complementary to the case of player in question choosing to develop the high profile module first.

A necessary condition for $T_1(\theta)$ to be an equilibrium is that it is optimal for type θ to follow the recommended equilibrium strategy $T_1(\theta)$ rather than mimic any other type θ'. That is, the partial derivative of $U(\theta,\theta')$ with respect to θ' has to be zero at $\theta'=\theta$. The partial derivative is

$$\frac{\partial U(\theta,\theta')}{\partial \theta'} = \left[2f(\theta')\left[1-F(\theta')\right]\right]\left[T_1(\theta)-U_H(\theta)\right]-\left[1-F(\theta')^2\right]T_1'(\theta)$$
$$-cU_F(\theta)f(\theta')\left[1-F(\theta')+2f(\theta')\right]\left[1-F(\theta')\right]\left[S(\theta,\theta')-T_1(\theta')\right]$$

Evaluate then the partial derivative at $\theta = \theta'$, and set the derivative equal to zero to get

$$cU_F(\theta)f(\theta)\left[1-F(\theta)+2U_H(\theta)f(\theta)\right]\left[1-F(\theta)\right]-\left[1-F(\theta')^2\right]T_1'(\theta)=0$$
$$\Rightarrow$$
$$T_1'(\theta)=\left[cU_F(\theta)-2U_H(\theta)\right]h(\theta)$$

To characterize the equilibrium strategies, we still need to find a boundary condition to tie down the differential equation T_1' above. The usual way in static incentive problems is to analyze the participation constraints of high-cost types. Here, that method is not usefull, as participation constraint consists of a dynamic optimization between conceding and waiting, and as non-participation is here equal to waiting forever. Fortunately, here there is an other way. The optimal decision by player with the lowest type $\underline{\theta}$ is to concede immediately, as she knows that there is no other type θ' who, in an equilibrium, would do so before her. Waiting a small amount $t > 0$ costs her t without affecting her probability to free ride, i.e., to win V_F, nor without increasing the speed at which the "second stage" war of attrition game evolves. To minimize her costs she then leaves the waiting game immediately, so that $T_1(\underline{\theta}) = 0$. Thus, we have the equilibrium waiting time

$$T_1(\theta)=\int_{\underline{\theta}}^{\theta}\left[cU_F(\theta)-2U_H(\theta)\right]h(\theta)dt.$$

The equilibrium strategy can also be interpreted as follows. At each point in time, the marginal individual with type θ has to be indifferent between developing the module at the time the equilibrium calls for and waiting a bit more to wait that types between θ and $\theta+d\theta$ concede. The cost of waiting consists of the probability $2h(\theta)\cdot d\theta = 2f(\theta)/(1-F(\theta))\cdot d\theta$ that another player is going to concede during this time interval times the utility value of the high profile task $U_H(\theta)$ that is lost if someone else concedes in this interval. This cost of waiting has to be equal to the difference between $T'(\theta;\theta')\cdot d\theta$, the time needed for types between θ and $\theta+d\theta$ to concede when two players remain, and $t'(\theta)\cdot d\theta$ the time needed for types between θ and $\theta+d\theta$ to concede when three players remain. For these two effects to be equal, the rate at which types stop waiting and concede needs to be slower after an initial concession. Otherwise, there would be no benefit in waiting, and players would find it optimal to concede earlier than the equilibrium calls for. In the equilibrium, players increase their chance of getting

the higher prize by conceding early, but this is compensated by a larger waiting cost in the following subgame.

For these strategies to constitute an equilibrium, stopping times need to be increasing in types. Otherwise, the rule used to update beliefs would not be consistent. A sufficient condition for stopping times to be increasing is in Assumption 1:

Assumption 1. $cU_F(\theta) \geqq 2U_H(\theta)$ for all θ.

This condition requires that the private benefits drawn from the high-profile module are significantly more valuable than from the low-profile module. The intuition is that if the high-profile low-cost task has low enough net programming costs $C_H(\theta)$, agents will not wait and take the risk of performing the low-profile high-cost task, so they prefer to concede immediately for the high-profile module.

Proposition 6. There is at most one symmetric separating equilibrium in the game.

The reason behind the Proposition 6 is the same as that of Proposition 2: the difference between the expected surplus of any two types is uniquely determined by the incentive compatibility arguments, as discussed above. Since any agent's probability of being able to free-ride is uniquely fixed, the difference between the two types' waiting costs must also be uniquely determined. If there were two different equilibria specifying different waiting times $T_1(\theta)$ and $T_2(\theta; \theta_R)$ for a type θ, these two equilibria would yield different differences between the type's waiting costs for at least one pair of types, which would offer that type a beneficial unilateral deviation. Hence, the separating equilibrium in this game, if one exists, must be unique.

Note that the expected net benefits of waiting increase with the value of free riding. Then as the value of free riding increases, the longer it takes on average for someone to volunteer. This is self-evident from the formula of the length of a two-player war of attrition derived in Proposition 5 above. But this then implies that the first volunteer can influence the length of the "second stage" war of attrition by her choice of task. There exists a trade-off between the gross payoff coming from the choice of task and the waiting cost that is borne by waiting for one more person to concede. When do the agents prefer to choose the high profile rather than the low profile task?

Proposition 7. If

$$U_H(\theta) > \frac{\int_{\theta}^{\bar{\theta}} U_H(t) f(t) \left[1 - F(t)\right] dt}{\left[1 - F(\theta)\right]^2}, \text{ for all } \theta,$$

all types choose the high-profile rather than the low-profile module when they are the first to develop a module.

The interpretation of the Proposition 7 is the following. The left-hand side is the benefit of producing the high rather than the low-profile module. The right-hand side is value to the type θ of the expected reduction in the length of the "second stage" war of attrition, in which the remaining players are trying to free ride and choose the developer for the low-profile module.

When the condition in Proposition 7 is not satisfied, i.e., when the private benefits of the high-profile module are large enough, there are some types who prefer to volunteer for the low-profile task. This is to speed up the development process. Call these types "impatient". Impatient types find the opportunity cost of waiting the F/OSS project to be completed more important than the net costs of programming the low-profile module.

We can now summarize the discussion above as Proposition 8.

Proposition 8. When the difference between the net development costs for different modules is large enough (Assumption 1 is satisfied) and nobody is impatient (the condition in the Proposition 7 is satisfied), the unique symmetric perfect Bayesian equilibrium is characterized by the waiting time $T_1(\theta)$ and the choice of the high-profile task by the first agent, and by the additional waiting time $T_2(\theta; \theta_R)$ in the subgame following the first decision by type θ_R

$$T_1(\theta) = \int_{\underline{\theta}}^{\theta} \left[cU_F(t) - 2U_H(t)\right] h(t) dt,$$
$$T_2(\theta; \theta_R) = c\int_{\theta_R}^{\theta} U_F(t) h(t) dt.$$

Here, "time is money" – time acts as a screening device, as high-cost types are willing to wait longer than low-cost types. Time takes the role of price system lacking from this "market", or the role of command mechanism or hierarchy lacking from this "organization", and acts as a coordinating device here. After the first F/OSS module has been released, high-cost types know they have a good chance to free ride. Thus, high-cost types are more willing to wait longer than low-cost types. This waiting game results in perfect sorting and is efficient for the task assignment. The player with the highest cost (or the

highest value for free-riding) is ready to wait the longest and gets to free-ride, and the agent with the lowest net cost develops the high-profile module. The inefficiency is due to the waste of time in F/OSS development.

7 Strategic Decision to Speed Up F/OSS Publishing

It may seem obvious that the first volunteer should always choose the high-profile task, as the low-profile module has a higher private net cost. However, this is not always true, as that decision need not minimize all the costs. The choice of low-profile module by the first volunteer reduces the costs faced by the second developer, and speeds up the "second-stage" two-player war of attrition subgame, hence the entire F/OSS project. I shall then look at the possibility that the first volunteer chooses the low-profile module in the this Section.

Consider the case in which the condition in the Proposition 7 is not satisfied for some types. This means that some types are impatient: they prefer to develop the low-profile module even when the high-profile one is available. The rationale here is that the loss in term of disutility of effort is more than compensated by the benefit of reducing the time waiting for the entire F/OSS project to be completed. To see the intuition, suppose the difference between the costs of the modules is very large, that there is a low-cost type that θ^l that receives high benefits from the completed F/OSS project $B(\theta^l)$ and two high-cost types that would wait for long before developing the low-profile module. Then if the player with θ^l chooses the low-profile module, the lower of the two types concedes in the waiting game more quickly for high-profile rather than low-profile module.

In equilibrium, the rate at which types concede depends on the hazard rate and on the patience of the types that are supposed to concede at that time. On might guess that low-cost types are more likely to be impatient. Suppose then that there exists a type θ_p such that types equal or lower than that are impatient and higher types are patient.

Proposition 9. When the heterogeneity of modules is large enough (Assumption 1 is holds) but some types are impatient (condition in Proposition 7 does not hold for types $\theta \leq \theta_P$), the unique symmetric Bayesian Nash equilibrium is characterized by a stopping time $T_1(\theta)$ and the choice of low-profile module for impatient type and the choice of the high-profile module for patient types,

$$T_1(\theta) = \int_{\underline{\theta}}^{\theta} \left[U_F(t) + U_H(t)\right] h(t)dt \text{ for } \theta \leq \theta_P, \text{ and}$$

$$T_1(\theta) = \int_{\underline{\theta}}^{\theta_P} \left[U_F(t) + U_H(t)\right] h(t)dt + \int_{\theta_P}^{\theta} \left[cU_F(t) - 2U_H(t)\right] h(t)dt \text{ for } \theta > \theta_P,$$

and an additional stopping time

$$T_2\left(\theta;\theta_R\right) = c\int_{\theta_R}^{\theta} W(t)h(t)dt, \text{ with } W(t) = U_F(t) \text{ or } W(t) = U_F(t) - U_H(t),$$

in the subgame following a decision to volunteer by type θ_R.

It can then be rational to choose the less attractive module even if a more attractive one is available. This happens when an individual has a high opportunity cost of waiting for the complete F/OSS package relative to the cost difference between tasks. When what matters is not the reputation, signaling or other such private benefits but to have the F/OSS project completed as early as possible, it can be an optimal strategy to volunteer for the low-profile task.

9 Pooling Equilibrium

So far, I have examined monotonic separating equilibrium in which waiting time coordinates agents' behavior. Here I turn to the case where the monotonic equilibrium does not exist. When the net costs of the high-profile module are very low, i.e., the private benefits are very high, relative to those of the low-profile module, choosing the high-profile module becomes very attractive. The problem then presents the characteristics of the Grab the Dollar game, and players have strong incentives to concede immediately.

In a pooling equilibrium, each player attempts to seize the opportunity to develop the high profile module, one of them is awarded the high-profile task, and the two remaining players enter in a war of attrition to decide who develops the low-profile module.[19] Since all types behave in the same way, players do not learn anything in the first stage of the game, and there is no updating of beliefs.

A player gets to develop the high-profile module with probability of 1/3, and then waits the expected length of the "second stage" game. With probability of 2/3, another player

[19] This rush can also be limited to low-cost types who volunteer immediately while higher types wait. I shall not analyze this semi-pooling equilibrium here.

gets the opportunity, and the player receives her type's expected surplus in the two-player "second stage" game. Her expected payoff is then

$$V_P(\theta) = \frac{1}{3}\left[U_H(\theta) - c\int_{\underline{\theta}}^{\bar{\theta}} U_F(t)f(t)(1-F(t))dt \right] + \frac{2}{3}\int_{\underline{\theta}}^{\theta} U_F'(t)F(t)dt.$$

The only deviation available is not to volunteer immediately. This yields an expected payoff equal to the surplus in the two-player war of attrition with probability 1

$$S(\theta) = \int_{\underline{\theta}}^{\theta} U_F'(t)F(t)dt.$$

Hence, for a pooling equilibrium to exist, the incentive constraint that the expected benefits from the development of the high-profile module at the first opportunity plus the expected surplus from the two-player "second stage" game exceeds the surplus from the two-player war of attrition probability 1 for all θ needs to be satisfied. This can be rewritten as Assumption 2:

Assumption 2. For all θ,

$$U_H(\theta) \geq \int_{\underline{\theta}}^{\theta} U_F'(\theta)F(\theta)dt + c\int_{\underline{\theta}}^{\bar{\theta}} U_F(\theta)f(t)\left[1-F(t)\right]dt.$$

Proposition 10. When Assumption 2 is satisfied, the unique symmetric Bayesian Nash equilibrium is characterized by the stopping times $T_1(\theta)$ and $T_2(\theta)$ for the high-profile and the low-profile task,

$$T_1(\theta) = 0,$$
$$T_2(\theta) = c\int_{\underline{\theta}}^{\theta} U_F(\theta)h(t)dt,$$

respectively.

For a rush to start, the high-profile module must be very valuable compared to free riding. Assumption 2 hints that such an equilibrium arises when the private value of the high-profile module is large, the expected cost of waiting until the two-player war of attrition is resolved is small, e.g., if the parameter *c* near zero, or if the surplus in the two-player war of attrition is small, ceteris paribus.

9 Conclusions

I have analyzed programmers' incentives to produce a F/OSS module instead of free-riding as a game of war of attrition. Even if each agent is tempted to wait for someone else to produce a piece of F/OSS, it may still be optimal to volunteer early for a high-profile module that creates reputation and signals programming ability. It may also be optimal to volunteer strategically for a low-profile module even if a high-profile module is available to speed up the process and to reduce the total costs of waiting.

The model provides a partial explanation for why F/OSS is being produced and by whom. The individuals who gain most from the F/OSS relative to their programming costs are most eager to produce F/OSS modules, and the individuals who have high programming costs relative to private and public benefits from software are willing to wait the longest. The main insight here is an explanation for how large F/OSS projects can be coordinated without markets or hierarchies such as firms. Here, "time is money" – impatience and opportunity costs guide the decisions of the agents, and substitute for lack of prices and formal hierarchical control.

The model predicts that only those potential F/OSS projects are completed that have high value. High-value projects are those in which all the essential modules have low enough private net programming costs, so that a developer is found for each module, or ones that have high enough public benefits, so that somebody is willing to develop also the high-cost modules to complete the project. If the public value of a complete F/OSS package is high for some individual, relative to her programming costs, that individual has an incentive to also produce those software modules that have low private value, e.g., that do not create reputation among hackers or provide valuable signal to labor or capital markets. Then also these low-profile tasks are taken care of.

Policy conclusions that can be drawn from the analysis are that those individuals or organizations that wish to support the completion of a F/OSS project should volunteer for low-profile modules, or increase the net private value of the development by subsidizing costs or other means. Rate of volunteering can also be increased by raising the associated private benefits. This can be accomplished by increasing reputational benefits drawn from F/OSS modules or from potential business opportunities built on the use of F/OSS.

Appendix

Proof of Lemma 2

Proof: The condition implies $B(\theta,t) > C(\theta)$ for all t. Since $B(\theta,t)$ is monotone and falling in t, $B(\theta,0)$ maximizes utility.

Proof of Proposition 1

Proof: A higher-cost type cannot choose to develop a module before a lower-cost type of the same player has done so. If the low-cost type gets the same expected surplus from strategies with two different probabilities of being a free-rider, the high-cost type strictly prefers the strategy with higher probability of free riding than the low-cost type.

Also, at no moment of time does any player develop a module with strictly positive probability. By symmetry, all players would have strictly positive probability of programming, but then any player would strictly prefer programming just after this point in time. So, $T_2()$ is strictly increasing in θ for any θ_R, and a player is able to free ride if and only if the other remaining player has type lower than her's. QED[20]

Proof of Proposition 3

Proof: Given that all other agents use this decision rule, the expected surplus of a type θ using the strategy of type θ' is

$$U(\theta,\theta') = \left[1 - F(t)\right]T_2(t) + \int_{\underline{\theta}}^{\theta'} f(t)\left[W(\theta) - T(t)\right]dt$$

in which the first term represents the utility of the agent θ when he is the first to develop the module, and the second term the expected utility of the player in case he is able to free ride. A necessary condition for an equilibrium is that $U(\theta,\theta') = 0$ at $\theta' = \theta$.

$$1 - F(t)T_2'(t) + f(t)W(\theta) = 0$$
$$\Rightarrow$$
$$T_2'(t) = \frac{f(\theta)W(\theta)}{1 - F(\theta)} = W(\theta)h(\theta)$$

The second-order condition is satisfied since $sign(\partial U(\theta,\theta')/\partial\theta) = sign(\theta - \theta')$. Using the boundary condition $T_2(\underline{\theta}) = 0$ we get the statement in the Proposition. QED[21]

Proof of Proposition 4

Proof: By definition, $S(\theta) = max_{\theta'} EU(\theta,\theta') = EU(\theta,\theta)$. Since all the functions are continuously differentiable, we can use the Envelope Theorem, and get

---

[20] See also Lemma 1 in Bulow and Klemperer. Note that this proof extends to $T_1()$ as well.

[21] See also Lemma 3 in Bulow and Klemperer.

$$S'(\theta) = EU(\theta,\theta) = \int_{\underline{\theta}}^{\theta} W'(t)f(t)dt = W'(\theta)F(\theta)$$

Integration yields

$$S(\theta) = \int_{\underline{\theta}}^{\theta} W'(t)F(t)dt + S(\underline{\theta}).$$

$S(\underline{\theta}) = 0$ as the lowest cost type does not wait. QED.

Proof of Proposition 5

Proof: This proof is an application of the Revenue Equivalence Theorem for auctions. A war of attrition is an optimal auction – the prize always goes to the highest type and that the surplus of the lowest type is zero. The expected cost per player in the war of attrition is exactly the expected duration of the war, as the cost of waiting is one per unit time. The expected cost per player is then equal to half the expected price paid by the winner of a second price auction. The price paid by the winner in the second-price auction is the expected value of the smaller bid or the expected value of the minimum of the players' valuation:

$E[R] = E[min(W(\theta_1),W(\theta_2))] = E[R] = 2ET =>$

$T = E[min(W(\theta_1),W(\theta_2))]/2$. QED.

Proof of Proposition 6

Proof**:** Use Lemma 2 of Bulow and Klemperer (1999).

Proof of Proposition 7

Proof: The payoff of a player of type θ who chooses to develop the high-profile module is

$$U_H(\theta) - \int_{\theta}^{\bar{\theta}} U_F(t)f(t \mid t > \theta)\left[1 - F(t \mid t > \theta)\right]dt.$$

The payoff of a player who chooses chooses to develop the low-profile module is

$$-\int_{\theta}^{\bar{\theta}} \left[U_F(t) - U_H(t)\right]f(t \mid t > \theta)\left[1 - F(t \mid t > \theta)\right]dt.$$

The condition then becomes

$$U_H(\theta) > \int_{\theta}^{\bar{\theta}} U_H(t)f(t \mid t > \theta)\left[1 - F(t \mid t > \theta)\right]dt$$

$$= \frac{\int_{\theta}^{\bar{\theta}} U_H(t)f(t)\left[1 - F(t)\right]dt}{\left[1 - F(\theta)\right]^2}.$$ QED.

Proof of Proposition 9

Proof: The derivation of $T_1(\theta,\theta')$ comes from Proposition 3. The expected utility of an agent with type θ behaving like a type θ' when the other agents follow the equilibrium strategy is

$$-\left[1-F(\theta')\right]^2 T_1(\theta') + c\int_{\theta'}^{\bar{\theta}} f(t)\left[1-F(t)\right]\left[U_F(t)-U_H(t)\right]dt$$
$$+\left[1-\left[1-F(\theta')\right]^2\right]U_H(t) + \int_{\theta'}^{\bar{\theta}} 2f(t)\left[1-F(t)\right]\left[S(\theta,t)-T_1(t)\right]dt$$

A necessary condition for T to be an equilibrium is that it is optimal for type θ to volunteer at time T(θ) rather than mimic type θ'. The derivative of the preceding expression with respect to θ' is

$$2f(\theta)\left[1-F(\theta')\right]T_1(\theta') - \left[1-F(\theta')\right]^2 T_1'(\theta') + cf(\theta')f(t)\left[1-F(\theta')\right]\left[U_F(t)-U_H(t)\right]$$
$$+2f(\theta)\left[1-F(\theta')\right]U_H(t) + 2f(\theta)\left[1-F(\theta')\right]\left[S(\theta,\theta')-T_1(\theta')\right].$$

This has to be zero when evaluated at θ'= θ, i.e.,

$$\left[1-F(\theta)\right]^2 T_1'(\theta) = cf(\theta)\left[1-F(\theta)\right]\left[U_F(\theta)-U_H(\theta)+2U_H(\theta)\right]$$
$$\Rightarrow$$
$$T_1'(\theta) = h(\theta)\left[U_F(\theta)+U_H(\theta)\right].\ \text{QED}$$

References

Bessen, J. (2002), "Open Source Software: Free Provision of Complex Public Goods", mimeo, Research on Innovation, available at http://www.researchoninnovation.org.

Bilodeau, M. and A. Slivinski, (1996), "Toilet Cleaning and Department Chairing: Volunteering for a Public Service," *Journal of Public Economics*, 59: 299-308.

Bitzer, J. and P.J.H Schröder (2003), "Bug-Fixing and Code-Writing: The Private Provision of Open Source Software", mimeo, DIW Berlin – German Institute for Economic Research. Paper presented at EARIE 2003.

Bliss, C. and B. Nalebuff (1984), "Dragon-Slaying and Ballroom Dancing: The Private Supply of a Public Good," *Journal of Public Economics*, 25: 1-12.

Bolton, P. and J. Farrell (1990), "Decentralization, Duplication and Delay," *Journal of Political Economics*, 98: 803-26.

Bulow, J. and P. Klemperer (1999), "The Generalized War of Attrition," *American Economic Review*, 89: 175-89.

Farrell, J. and G. Saloner (1988), "Coordination Through Committees and Markets," *Rand Journal of Economics*, 19: 235-53.

Hendricks, K., A. Weiss and C. Wilson (1988), "The War of Attrition in Continuous Time with Complete Information", *International Economic Review*, 29: 663-80.

Johnson, J.P. (2002), "Economics of Open Source Software," *Journal of Economics & Management Strategy*, 11: 637–662.

Klemperer, P. (2003), "Why Every Economist Should Learn Some Auction Theory", in M. Dewatripont, L. Hansen and S. Turnovsky (eds.), *Advances in Economics and Econometrics: Theory and Applications*, Vol. 1, pp. 25-55, Cambridge University Press.

Lee, S., N. Moisa and M. Weiss (2003), "Open Source as a Signalling Device - An Economic Analysis", Working Paper No. 102, Goethe-University, Frankfurt am Main.

Leppämäki, M. and M. Mustonen (2003), "Spence Revisited - Signalling with Externality: The Case of Open Source Programming", mimeo, University of Helsinki.

Lerner, J. and J. Tirole (2002), "Some Simple Economics of Open Source", *Journal of Industrial Economics*, 52: 197–234.

Morgan, J. and V. Krishna (1997), "An Analysis of the War of Attrition and the All-Pay Auction", *Journal of Economic Theory*, 72: 343-362.

Mustonen, M. (2002), "Why Do Firms Support the Development of Substitute Copyleft Programs", Discussion Paper No 439, University of Helsinki.

Mustonen, M. (2003), "Copyleft – The Economics of Linux and other Open Source Software", *Information Economics and Policy*, forthcoming.

Myerson, R. (1981), "Optimal Auction Design," *Mathematics of Operation Research*, 6, 58-73.

Park, A. and L. Smith (2003), "Caller Number Five: Timing Games that Morph from One Form to Another", mimeo, University of Cambridge, available at http://www.econ.cam.ac.uk/phd/ap248/

Ponsati, C., and J. Sákovics (1995), "The War of Attrition with Incomplete Information," *Mathematical Social Sciences*, 29, 239–54.

Raymond, E.S. (1999), *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Associates, Inc., available at http://www.catb.org/~esr/writings/cathedral-bazaar/.

Riley, J. and W. Samuelson (1981), "Optimal Auctions," American Economic Review, 71, 381-392.

Sahuguet, N. (2003), "Volunteering Public Services when Tasks are not Equivalent", mimeo, Université Libre de Bruxelles.

Smith, M.A. and P. Kollock, eds. (1999), *Communities in Cyberspace*, Routledge.

Weber, S. (2004), *The Success of Open Source*, Harvard University Press.