Competing communities of users and developers of computer software: competition between open source software and commercial software

by

Marc van Wegberg[1] and Peter Berends[2]

May, 2000

NIBOR working paper, NIBOR/RM/00/001

Abstract. The open source movement is a group of volunteer programmers that has recently caused quite a stir in the software market. The volunteers of this group develop computer operating systems, programming languages, and other software. They work together in teams that communicate via the Internet. Their goal is to develop useful software that is available for free and that users can change at will. To enable users to change and improve the software, they distribute their software not only in compiled form (that a computer needs to actually run the software), but also in its source code (the lines of code that programmers write). This cooperation breaks through the usual barriers that separate corporate suppliers from their buyers. This may represent the first example where the Internet enables cooperation on a scale that changes market dynamics. This paper studies the interaction between the network dynamics of the open source movement and the dynamics of a commercial software supplier. It makes our first step in identifying conditions that support a successful development of open source software. We focus on one particular set of projects within the open source movement, namely the Linux computer operating system.

[1] Faculty of Economics and Business Administration, University of Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands, tel. +31 43-3883654, fax. +31 43- 3884877, e-mail: M.vanWegberg@MW.UniMaas.nl, home page: http://www.unimaas.nl/~mwegberg/index.htm. We thank the participants of the '6th International Conference on Multi-organisational Partnerships and Cooperative Strategy' at WORC, Tilburg University, the Netherlands, for their comments.
[2] Faculty of Economics and Business Administration, University of Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands, tel. +31 43-3883622, e-mail: P.Berends@mw.unimaas.nl.

Introduction

The aim of this paper is to explore the reasons for computer users to volunteer in cooperating with each other to develop and improve the product they use. Many people use the Internet to develop and participate in networks of users and developers. The open source movement is a lively example, where thousands of programmers throughout the world interact with each other via the Internet to exchange ideas, experience, and lines of programming code. Commercial suppliers of software benefit from user feedback, which they use to test and improve their product. The open source movement goes one step further: it develops its own products, and uses user feedback to determine which direction to go in: user feedforward as well as user feedback. In direct competition with commercial software suppliers, the open source movement has strengths and weaknesses. We present a simulation model that highlights these, and indicates survival chances of both commercial and open source software.

The paper first dicusses the Linux operating system, an example of an open source cooperative project. It then discusses how networks such as the one around Linux emerge. An important question is why people participate in them. A simulation model identifies some factors that drive or impede an open source network from emerging. The paper concludes with a brief discussion of how the interaction between commercial software suppliers and the open source movement may be better understood.

A brief history of Linux

The open source movement consists of numerous programmers, who cooperate in many projects. These projects can be unrelated to each other. This paper focuses on the Linux operating system. One should keep in mind that although for the general public Linux is the best known open source program, there are many others.

Linux is a Unix clone. Unix is a heavy-duty operating system for professional, intensively-used, computers such as workstations and servers. AT&T originally developed Unix in the 1970s, and owns property rights on the software. Within the scientific community, programmers developed versions of Unix that were free from these property rights. An example is professor Andrew Tanenbaum at the Vrije Universiteit in Amsterdam. He developed a Unix clone Minix to help students learn about Unix. He introduced the program in January 1987. It is now freely available for download over the Internet from Andrew Tanenbaum's website (see appendix 2 with online sources). It is, however, not free software in the sense of the open source movement: its copyright rests with the publisher Prentice-Hall. People can use it, but they cannot change it.

In 1991, the Finnish student Linus Torvalds was dissatisfied with MS-DOS and the Apple Macintosh. He thought they were not stable and reliable enough. MS-DOS, moreover, did not utilize the full potential of Intel's new 386 processor. Linus Torvalds considered Unix a superior operating system that, however, was available only for expensive heavy-duty computers. So he developed a Unix clone. His motive was to develop a reliable, stable, operating system that used the full potential of the Intel 386 processor. He started with programming the barest essentials of an operating system. Initially, this was an offspring of the Minix program. Via an appeal in the comp.os.minix newsgroup, dated 5 Oct. 1991, Linus involved 'hackers' (as he calls them) in his project. This is probably, with hindsight, the start of the Linux project. He published the system that he programmed on the Internet. He published, that is, not only the operating system but also the source code, the lines of programming code that make up the software program. This set a snowball going of other people improving and contributing to his work. In 1997, Linus Torvalds migrated to the United States to work for a new company, Transmeta. This company is involved in some project they keep secret, but that relates to microprocessors. Linus Torvalds seems to have had very pragmatic objectives with this Unix clone: to learn more from the 386

microprocessor (source: Linux International website). So the Linux OS may have been, with hindsight, a side-effect of another project!

Many programmers volunteered to exchange their software via the Internet. The evolution of Linux would have been unthinkable without the Internet. It may represent perhaps one of the most important and earliest examples of how using the Internet changes the market place.

What made it easy for fellow programmers to contribute to the Linux project, is that the operating system is highly modular. It consists of separate programs, that fulfill individual roles of a computer operating system. One of these is the kernel, which directly addresses and controls the computer's hardware.[3] Programmers tend to have their specific needs. They write or rewrite a program to better suite their specific need, given the computer they use or the tasks they use it for. Via the Internet they exchange their programs, look into them, and improve or offer comments for improvement. Torvalds' task consisted for a large extent of coordination and motivation. He adopted the best programs into his operating system. And thus it grew.

Linux programmers and users may be said to form a community. Like any community, they need some culture, rules, and some institutionalisation. Linux grew too big for one person to handle. It is no longer the person Linus Torvalds who controls this community. Moreover, anyone can enter this community. Unlike a firm, selection and dismissals cannot control access to the community.

For users, the modularity of the operating system poses problems. Getting a working operating system requires one to download all kinds of files. These files are source code. They have to be compiled into working computer code.[4] For many users, this requires more expertise or time than they have. To make it easier for potential users to install Linux, commercial companies emerged. They sell Linux distributions. A distribution is a collection of source code and compiled software, that gives the buyer a fully operational operating system. There are several companies that sell a Linux distribution, both operating software and applications. They include Red Hat, Caldera, and S.u.S.e. The operating system they sell, and many additional software, can be downloaded free from the Internet. There are nevertheless people willing to pay money to these companies. The added value from commercial companies is that they offer a selection of software, combined in one package, often with software that helps to install the Unix software on one's computer. These companies reduce the transaction costs that buyers would otherwise incur by searching on the Internet for software. Since many potential users do not have the skill to combine their own software from public sources, these commercial companies serve as intermediaries.

Recent events suggest that 1998 may be the year where the business community discovered Linux. Software companies such as Informix and Oracle now adapt their software to the Linux platform. This is giving credibility and legitimisation to Linux.

Some of the supporters have an anti-Microsoft agenda. Oracle is constantly looking for ways to reduce its dependence on Microsoft. It developed the Network Computer as an alternative for Windows PCs and it embraced Java as a possible alternative operating system to Windows. Caldera belongs to Ray Noorda, who once led Novell into an anti-Microsoft course. This strategy ultimately failed, and Novell barely survived. It still exists, and is even profitable again, but it has lost its dominant position in software for enterprise networking. Windows NT and Unix threaten to overtake Novell's Netware as operating system of the network. With this history in mind, Caldera probably has an anti-Microsoft agenda too. It still markets a clone to the MS-DOS operating system (DR-DOS).

[3] Not every operating systerm has a separate kernel: having one is one of the hallmarks of a modular design.
[4] The source code consists of lines of text, programming codes. The actual software has to be compiled into the bits and bytes that the computer's hardware is able to understand.

Networks and markets

The Open Source movement is an extreme form of a networked, distributed, joint effort. A network, like a market, is a way to coordinate and cooperate in (economic) activities. Characteristic for a network is that all participants make a contribution. In contrast, a market has a sharp distinction between contributors (suppliers) and users (buyers). There are various roles that participants in a network can play. For example one can coordinate activities, contribute technological, financial inputs, or merely use others (free-riders). The contribution one makes to the network will affect one's position in the network. The reputation a participant in a network builds up reflects this position. The prospect of a reputation may motivate members of a network to participate to its activities. In a relational perspective on networks, reputation may be the main stimulus for making costly contributions to the network. In a performance-oriented perspective, people contribute because they expect the network to deliver outputs that they need.

A network can stimulate entry of new and participation by established members by improving the ability to build reputation (the social perspective), by increasing the output of the network (the performance perspective), or by reducing the costs of making a contribution. The Internet has made the important contribution of very strongly reducing the latter costs. For open source networks involving user feedback, the Internet is a necessary condition of existence.

If a network has ways to recognise and acknowledge a partner's contribution, it facilitates the development of reputation, and thus will elicit more contributions. Is reputation an important motivator for people? The answer may be negative, in a society where indiduals have life-long employment relations with an employer. In a networked freelance economy, however, sometimes called an e-lance economy, individuals need to develop a reputation in order to attract customers (formerly known as employers) for their work (see Malone and Laubacher, 1999, for the e-lance economy).

Whether we now really witness the dawn of an e-lance economy remains to be seen. What is clearly true is that individuals own the means of production needed to participate in an e-lance economy. Unlike in manufacturing industries, information services need a production tool (a networked PC) that almost anyone can afford. The Internet serves as an input in this system, as well as a low cost distribution and transport (of information products) system. The falling costs of hardware, software, content and networking underpin the ability of individuals to team up and create joint services. That the same instrument (a networked PC system) serves both in consumption and in production activities, may break down the traditional distinction between supply and demand, production and consumption, in a market economy.

Networked e-lancers change the competitive environment for integrated organisations. One reason for an interest in the software market, is to see how the interaction between integrated organisations and groups of e-lancers plays out. On the one hand we have here the most advanced and experienced group of e-lancers, the open source movement. On the other hand, we have one of the best managed, wealthiest corporations in the world, Microsoft.

That the open source movement makes users make contributions has an interesting parallel with distributors such as supermarkets (Wal-mart) and some retailers, such as Ikea. These companies sell cheap products and they expect buyers to do part of the work: bring the goods to the counter, transport them home, install the furniture themselves. The secret of this formula is that a little bit of self-help by a huge amount of users adds up to a massive cost saving by the supplier. What is new about the open source movement, is that it involves this self-help into the production process of the software.

If the partners work on a joint product, it matters how well coordinated their activities need to be. The higher the costs of mis-coordination, the more centralised in some way decision making in the network will tend to become. A modular system consists of various components that are semi-independent: they interact, but they do have areas of autonomy. The more modular a system or technology is, the higher the ability of a distributed team to develop or supply such a system or technology.

Modelling the adoption of software

Users do not only learn to use a computer system, they also give feedback to help each other. Why would users spend their time to give free help to others, many of whom they may not know in person? Collectively, users benefit from helping each other, but individually, a user may not want to share his knowledge if that takes time. Still, they do offer help. In newsgroups, people share experience. And in the open source movement, programmers volunteer to program code for expanding or improving a program. There are various arguments for this selfless behaviour.

Characteristically for the open source movement, programmers try to do justice to the contributions made by other programmers. This gives a sense of ownership and pride in the open source software, which is unique to this movement. Feedback builds up a personal reputation in the community of users and suppliers of a software program. That may offer an incentive for participating. Commercial software companies may pay people for feedback, as an alternative incentive. Feedback can also be a form of selfhelp. By publishing on the Internet how he has solved a problem, a programmer shares his ideas with others, which costs him little, and offers the potential for improvements by others. Giving feedback may also provide utility to the user from his anticipation of how his feedback improves product quality. Solving problems can also be fun, an intellectual challenge. For some, like perhaps for Richard Stalman of the Free Software Foundation, who pioneered open source software, open source software represents a political or social value, related to freedom of speech. In the longer term, these induced improvements will benefit the individual user. These reputational, problem solving, and improvement oriented kinds of user feedback tend to increase with the importance of the program to the users, the importance of improvements to them, the ease with which they can communicate their feedback, and the response by the other party to the feedback. We will focus here on the utility that users derive from a reputation when they contribute to improving the quality of the program. In our focus on reputation we folow the explanation that Raymond (1998) gives about the success of the open source model.

The model

A user derives utility from using a computer with a particular operating system. To use it, he does need to put in effort to learn about the software. This effort builds up experience over time. In later periods, experience reduces the need for learning effort. In other words, experience from past effort is a substitute for current effort. The user also provides feedback (f), which builds up a reputation (F), that is a source of (anticipated) utility to the user. The following utility function captures these effects:

$$(1) \qquad U(e_j,f_j) = Q_i h_j (e_j+X_{ji})^{\alpha} + a_j F(f_j)^{\gamma} - p_i - (e_j + f_j)w_j \ ,$$

where j is the user, i (= 1 or 2) is the operating system, $0 \le a_j$, $0 \le \alpha < 1$ and $0 \le \gamma < 1$. The parameter *w* (> 0) is the (possibly subjective) wage cost of effort *e*, *Q* is the quality of the program, *h* (> 0) is the individual user's talent in using computer software, *p* is the price of the program (which we separate from the price of computer hardware), *e* is the learning effort to make the program work, and *X* is the experience in using a computer operating system. For

the sake of modelling convenience, we treat experience *X* as a perfect substitute for effort. Using an operating system gives experience. The experience may come with the time that one uses the system, or it may increase with the effort one invests in using it. We follow the latter route:

(2a) $X_{ji} = (1-\delta_x)X_{ji,-1} + \varepsilon_x e_{ji,-1}$ where $0 \le \delta_x \le 1$, $0 \le \varepsilon_x \le 1$, if user j continues to use system i.

The user chooses a level of effort, e, in the trade-off between the direct advantage of effort, the first term on the right hand side of equation (1), and the wage cost *ew*. He (or she) ignores the indirect effect of effort, in that it builds up experience, which ceates future benefits. The user is myopic, in that he ignores future effects of his current decisions. We will assume that switching to another operating system leads to a loss of some experience, but not necessarily all:

(2b) $X_{jk} = \lambda_x X_{ji}$, where $0 \le \lambda_x \le 1$, if the user switches from system i to system k.

The switching cost implicit in the factor λ_x represents the lock-in effect, that a user feels compelled to continue using his initially preferred system. The experience level is thus endogenous in the limited sense that by the decision to switch, the user will reduce his (effective) experience level.

The utility factor $aF(f_j)^{\gamma}$ captures the reputation motive for user feedback. If some people care more about reputation that others, this may turn up in the parameters a or γ. Reputation is the outcome of providing feedback, which is the activity f_j. The anticipated reputation $F(f_j)$ equals the following expression:

(3a) $F(f_j) = (1-\delta_f)F_{-1} + \varepsilon_{fi}N_i h_j X_{ji} f_j$,

where N_i is the size of the network where the user builds up a reputation (taken to be the known number of users of the program *i* from the previous period). Both experience (X) and talent (h) are required to give the kind of feedback that creates a reputation. We may call the expression $h_j X_{ji} f_j$ the *effective* feedback. The factor $\varepsilon_{fi}N_i$ indicates the ability of the network to acknowledge feedback and build a valuable reputation. Reputation wears off over time (the factor δ_f). If the open source movement and the commercial software differ in the extent to which they enable users to build up a reputation, the parameter ε_{fi} may differ between the two programs. If a user switches to another operating system, he loses much of his reputation. We capture this switching cost by the parameter λ_f, where $0 \le \lambda_f \le 1$, and

(3b) $F(f_{jk}) = (1-\delta_f)\lambda_f F_{ji,-1} + \varepsilon_{fi}N_k h_j X_{jk} f_{jk}$,

where j is the user and k the operating system he switches to (from system i).

The adoption level of an operating system decreases relative to the previous period by those users that switch, and it increases with new users and those who switch from the other system. Each period, a potential user chooses between four options: continuing not to use a computer, continuing to use their current system, buying the new version of their current system, or switching to the alternative system. This gives five possible outcomes: no PC (situation 1), hold system 1 (situation 2), hold system 2 (situation 3), buy 1 (situation 4), and buy 2 (situation 5). Call the set of buyers of a system B_i (for systems 1 and 2). Call A_i the set of adopters of system i.

Quality may decrease by obsolescence due to viruses or new environmental demands. This tends to reduce the quality of an existing system:

(4a) $Q_i = (1-\delta_i)Q_{-i}$, where i = 1 or 2.

The motive to stay with the current system rather than buying a new version of the same program is that this does not require the user to pay the price p_i of a new version. The disadvantage of holding on to the current system consists of the direct effect, that quality decreases due to obsolescence, as well as of the indirect effects that lower quality induces a lower level of effort. These indirect effects reduce the levels of experience and, in a later stage, reputation.

The new versions of the operating systems also suffer from obsolescence, but they also benefit from quality enhancing processes. These are user feedback and investments to improve the quality of commercial software. The adopters of a program (the set A_i) provide user feedback:

(5) $E = e_i Ó_{Ai} h_j X_{ji} f_j$, where i = 1 or 2, and $h_j X_{ji} f_j$ is the effective feedback per user.

For convenience, we assume that $e_1 = e$, and $e_2 = 1$. If $e > 1$, the open source movement succeeds better than commercial software in getting effective user feedback. The supplier of commercial software incurs an investment I_2 to increase product quality by $\nu_2 I_2^{½}$ in the next period. The commercial investment I_2 may consist of solicited user feedback, with ν_2 representing the price required to pay for this feedback. The supplier chooses the investment level with a view to expected future profits. The expections may in turn depend on current profits. A simplified investment function then may link today's investments to today's profits as follows:

(6) $I_2 = \beta(p_2-c_2)Ó_{B2}$,

where $Ó_{B2}$ is the number of buyers, p_2 is the price of a unit of system software, and c_2 its (distribution) cost. This represents adaptive behaviour by the commercial supplier. These actions determine quality:

(4b) $Q_1 = (1-\delta_1)Q_{1,-1} + \max\{0, \sqrt{(E_{-1}-g_1)}\}$,

and

(4c) $Q_2 = (1-\delta_2)Q_{2,-1} + \nu_2\sqrt{I_{2,-1}} + \max\{0, \sqrt{(E_{-1}-g_2)}\}$,

where $g_i, \delta_i > 0$ (for i = 1 or 2) and $\nu_2 > 0$.

Given the utility function defined by equations (1) to (3), the user chooses an effort (e) and a feedback (f) that maximizes utility. The first order conditions for maximization are:

(7) $\partial U(e,f)/\partial e = 0$, which gives $\alpha Qh(e+X)^{\alpha-1} - w = 0$

that is,

$$(8)\quad e = \begin{cases} -X+(w/\alpha Qh)^{1/(\alpha-1)}, \text{ if } \alpha > 0,\ Q > 0,\ h > 0, \text{ and } X < (w/\alpha Qh)^{1/(\alpha-1)}, \\ 0, \text{ otherwise.} \end{cases}$$

If effort is zero, the user's experience decreases to $X_{ji} = (1-\delta_x)X_{ji,-1}$ (if he does not switch). The user also chooses a level of feedback:

(9) $\partial U(e,f)/\partial f = 0$, which gives $\gamma a \varepsilon_f NhX((1-\delta_f)F_{ji,-1}+\varepsilon_f NhXf)^{\gamma-1} - w = 0$,

If $\gamma a \varepsilon_f NhX > 0$ this gives:

$$(10) \quad f_{ji} = \begin{matrix} (-(1-\delta_f)F_{ji,-1} + ({}^{w}/(\gamma a \varepsilon_f NhX))^{1/(\gamma-1)})/(\varepsilon_f NhX) \\ 0, \text{ which ever is larger.} \end{matrix}$$

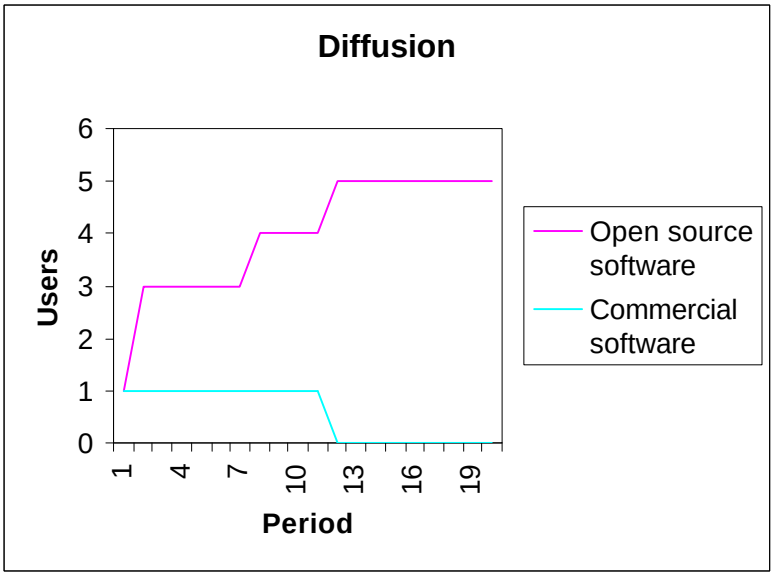
If $\gamma a \varepsilon_f NhX = 0$, then $f_{ji} = 0$. This feedback gives a reputation:

$$(11) \quad F_{ji} = \begin{matrix} ({}^{w}/(\gamma a \varepsilon_f NhX))^{1/(\gamma-1)}, \text{ if } f_{ji} > 0, \\ (1-\delta_f)F_{ji,-1}, \text{ otherwise.} \end{matrix}$$

In the case where the user holds his current operating system, equation (4a) defines the quality, as it degrades over time. The utility function is equation (1), but with a price of zero as the user already owns the operating system. Equation (2a) defines his experience and equation (3a) his reputation. If the resulting optimal choice of effort and feedback give a utility higher than zero (not using a computer) and higher than the utility of buying a new system, then the user will continue to hold on to his system, rather than buying a new version.

Simulation results

We simulated the model using various parameter constellations. See the appendix for one particular setting. Figure 1 shows the associated outcome of the competition. This represents a case where the open source software corners the market, the market grows until all potential users buy software, and commercial software drops out. Depending on the parameters, other outcomes occur as well.



We may also vary some parameters. We look at two in particular. One is ε_{f1}. This represents the extent to which a user of open source software is able to develop a reputation for high quality feedback. The other is the initial condition: which users pioneer the open source movement? We simulate with five users. Their only difference is in the talent for programming. In all simulations reported, user nr. 1 (with the lowest talent) starts with commercial software. There is also one initial user of open source software, who ranks second, third, etc., up to fifth (with increasing talent). The other three potential users, start without using a computer. This allows us to simulate market growth. We capture the initial condition by measuring the talent of the one pioneer user of open source movement (h_1). The table summarizes the outcome by the average number of users of the open source program, over the five year period (periods 9 to 13).

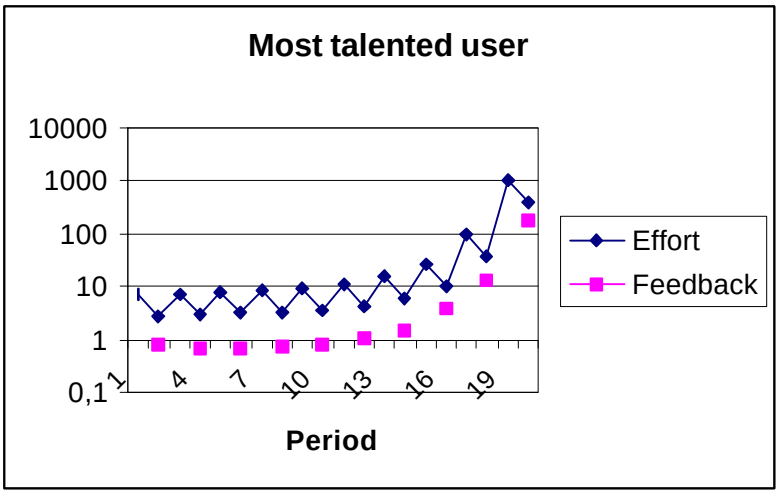
Pioneers (h_1)	0,9	3	3	3	3	4,4
	0,7	3	3	3	3	3,8
	0,5	0	0	0	0	0
	0,3	0	0	0	0	0
		0,1	0,3	0,5	0,7	0,9
		Reputation (ε_{f1})				

Table 1 Simulation results

The table illustrates that an increase in pioneering talent tends to increase the long-term number of users of open soure. In other words, the number of pioneers weighted with their talent determines diffusion. This illustrates the meritocratic approach of the open source. It is not the market share of open source software that determines its future viability, but its share among the talented users, that counts. The table also illustrates (slightly) that the better the open source movement succeeds in building a reputation for talented users (the factor $å_{f1}$), the higher its long term diffusion tends to be. The open source movement harnesses the selfish motive of creating a reputation within a group to the task of developing software.

The simulations also display other features. As the graph above shows, it is possible that a group (here, one person) uses a system for a long time, without anyone dropping out or joining the program's user base. This stickiness result from the fact that a user can buy a program, and then continue using it for a long time. An existing program loses quality in absolute sense (which we explained by viruses), and it also loses compared to the new versions, which benefit from improvements. An existing program is also, however, free to the user, while a new version costs money. This causes the stickiness that a program, which may no longer be developed (that is, the simulation records an absence of feedback being given) can still survive for some periods.

A feature of the model (which the simulations, of course, also exhibit) is that if the user base drops to zero, a program cannot recover. This irreversibility is due to the fact that if the user base drops to zero, the ability to build up a reputation is lost, and no user will be interested in providing feedback. Hence the route to quality improvement is closed off. This account cannot, therefore, exlain how a network emerges. To explain this we need arguments which, if not outside of the realm of economics, are at least outside the realm of our model.



At individual level, as the figure shows, the increasing quality of the software induces an increasing level of effort and feedback. This means that the quality increase itself also tends to accelerate. It does suggest that the model underestimates the effort required to develop the software program in its initial stages, when quality is low, but perhaps, the scope for developing a reputation is greatest.

The model has a knife edge character. For example, in simulation 2, the case where ν_2 equals 1,4 has as outcome that all five users adopt the open source model, but if ν_2 increases just to 1,5, then all five users switch to the commercial program. The larger effectiveness of an investment in the commercial program (ν_2 in equation 4c) tilts the balance. This feature of the model owes much to the importance of network size, N, to the reputation creation process (in equation 3) and thus the incentive to give feedback. The model exhibits a kind of natural monopoly: for talented users who want to build a reputation, it is best if all users adopt one program.

In simulation 3, we focus on the effect of initial conditions. We look at the talent of the pioneer user of the open source software (h_i) and at the initial investment in quality by the commercial supplier, $I_{2,0}$. See table 2:

Investment	2	0	0	0	0
	1,5	0	0	0	5
	1	0	0	5	5
	0,75	0	4,6	5	5
		0,3	0,5	0,7	0,9
	Talented open source pioneer				

Table 2 Simulation results

Table 2 makes the point that both the open source movement and the commercial supplier face a game that is determined at t = 0: the initial investment by the commercial supplier, and the number of open source pioneers (weighted with their talent) determine the outcome of the competition. What neither party will know, at the initial point, is whether this particular situation will let the one survive or the other. They do know (from table 2) that better initial conditions improve their survival chance. As the simulated outcomes show, the open source approach tends to either fail completely (zero users in runs 10 to 14) or to dominate entirely (close to five users, where it covers the market completely). Why would the open source program benefit from users with a low talent? The reason is two-fold. First, a large number of users with low talent and little relevant feedback can add up to significant overall feedback. And, these low talented users have an indirect effect: they enlarge the network, which increases the incentives for giving feedback (see equation 3). Hence the most talented users

give more feedback, when they know that more people use the program. This feature of the model raises the question whether a niche approach, as for example sought by Apple for its Macintosh operating system, can be viable if user feedback is an important feature of the competition.

Conclusions

This simple model is an attempt to come to terms with the unexpected development of the open source movement. It highlights some features of this movement, in particular, that it is based on activities undertaken by the users themselves. Our model highlights one motive for these activities: to build up a reputation in a network of users. It seems relevant to explore other motives, such as political views, altruism, or gift exchange, where programmers help each other in a quid pro quo, etc. Another simplification in the model is that it is about competition between the open source movement and a commercial supplier of software. Recent events show that many companies try to cooperate with the open source movement. These companies want to find out the reason for the success of this movement, in order to integrate some parts of that in their business model.

References

Frey, Bruno S. (1993) Shirking or Work Morale? The Impact of Regulating, *European Economic Review* 37: 1523-1532.

Grant, Robert M. (1996) Towards a Knowledge-based Theory of the Firm, *Strategic Management Journal*, 17 (Winter Special Issue), 109-122.

Malone, Thomas W. and Robert J. Laubacher (1999) The Dawn of the E-lance Economy, in: August-Wilhelm Scheer and Markus Nüttgens (eds.) *Electronic Business Engineering*, Heidelberg: Physica-Verlag, p. 13-24.

Raymond, Eric S. (1998) *The Cathedral and the Bazaar*, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/index.html

Appendix 1 Simulation 1

The tabel shows the parameters used for the reported simulation 1:

Parameters								
Both products	δ_f	0,3	Product 1	w_1	1	Product 2	w_1	1
				p_1	1		p_2	2
	λ_f	0,4		e	2			
	a	0,9		δ_1	0,2		δ_2	0,2
				g_1	0,1		g_2	0,1
							ν_2	1
							β	0,5
							c_2	0,3
				α_1	0,6		α_2	0,6
				γ_1	0,5		γ_2	0,5
				ε_{f1}	0,9		ε_{f2}	0,1
t=0				$Q_{1,0}$	4		$Q_{2,0}$	4
							$I_{2,0}$	2
All users	λ_x	0,4						
	δ_x	0,5						
	ε_x	0,2						
Users	Nr.	1	2	3	4	5		
talent	h_i	0,1	0,3	0,5	0,7	0,9		
t=0	System	2	0	0	0	1		
	experience (X)	1	1	1	1	1		
	feedback (f)	1	1	1	1	1		
	reputation (F)	1	1	1	1	1		
	effective feedback	0,1	0,3	0,5	0,7	0,9		

Appendix 2: Online sources

It won't come as a surprise that many information sources over Linux are online. We drew on the following one:

Linux distributors, both commercial and not-for-profit
Caldera, http://www.caldera.com/
Debian, http://www.debian.org/
Red Hat, http://www.redhat.com/
S.u.S.e, http://www.suse.com/
VA Research, supplies PCs that use Linux, http://www.varesearch.com/

FTP sites contain the files that make up the Linux operating system, including manuals, for downloading (e.g., nic.funet.fi, the /pub/OS/Linux directory, from a 1996 reference)

General ICT information sources
Automatisering Gids, paper-based and online Dutch IT journal, http://www.automatiseringgids.nl/
C:Net, online American journal and software distribution website,
http://www.cnet.com/?st.co.gp.rb.cn
Computable, paper-based and online Dutch IT journal, http://www.computable.nl/
Infoworld, paper-based and online American IT journal, http://www.infoworld.com/

Information sources specifically for Linux
Linux Gazette, news of the Linux world, http://www.ssc.com/lg/
Linux HQ, http://www.linuxhq.com/
Linux Online, managed by Michael McLagan, http://www.linux.org/
Slashdot, http://slashdot.org/index.shtml

Institutions (non-profit) in and around the Linux community (e.g., which specify licensing formulas):
Free Software Foundation, linked to GNU, for 'copy-left' licensing
GNU, created by Richard Stallman, for an open source Unix clone, http://www.gnu.org/
Linux International, promotion for Linux, http://www.li.org/
Open Source Initiative (OSI), created by Eric S. Raymond, not a membership organisation,
http://www.opensource.net/
Software in the Public Interest, Inc., a non-profit umbrella organisation that runs various GNU projects, including Debian, a non-profit Linux distributor, http://www.spi-inc.org/

Mailing lists
<dozens of 'em>

Newsgroups
<dozens of 'em>
A mail by Linus Torvalds on in 1991 in the newsgroup *comp.os.minix* (which still exists) started the Linux snowball.
A newsgroup like *comp.os.linux.help* shows how newsgroups work like helpdesks. An old one is *comp.os.linux*. A very active one and rather chatty is *comp.os.linux.advocacy*.

People
Tim Berners-Lee, inventor of the WWW, http://www.w3.org/People/Berners-Lee/
Donald Knuth of Stanford University, developer of TeX, http://www-cs-faculty.stanford.edu/~knuth/
Eric S. Raymond, analyst and proponent of the Open Source movement, http://www.tuxedo.org/~esr/
Richard Stallman, pioneer of the GNU organisation and the Free Software Foundation,
http://www.gnu.org/people/rms.html
Andrew S. Tanenbaum, creator of Minix, http://www.cs.vu.nl/~ast/
Linus Torvalds, creator of Linux, see http://linuxresources.com/linus.html for information about him

Andrew Tridgell, wrote Samba, see http://samba.org/~tridge/
Larry Wall, maker of the Perl language

Software distributed (sold or offered for free) as open source code
Apache server, see http://www.apache.org/
Berkeley Internet Name Daemon (BIND)
BSD (free BSD or Open BSD), see http://www.freebsd.org/index.html
Gnome, GNU Network Object Model Environment, GUI for Linux, http://www.gnome.org/
GNUstep, GUI and object-oriented programming environment for Linux, http://www.gnustep.org/
GTK, The GIMP Toolkit, software libraries that programmers can use for Linux programming, http://www.gtk.org/
InterNet News (INN)
KDE, the K Desktop Environment, GUI for Linux, http://www.kde.org/
Linux operating system
Netscape browser (since 1998), from http://www.mozilla.org/
Perl language, created and controlled by Larry Wall, see http://www.perl.com/pace/pub
Samba, by Andrew Tridgell, turns a Unix machine into a file and printer server, see http://uk.samba.org/samba/team.html
Sendmail, see http://www.sendmail.com/
TeX, type-setting language for mathematical text, see the user group http://www.tug.org/

Standard setting organisations
Linux Standards Association (LSA) created by Michael McLagan in 1998
Linux Standard Base, created in response to the LSA, http://www.linuxbase.org/