1-1-2000

Open Source Movements as a Model for Organizing

Jan Ljungberg
Viktoria Institute, janl@viktoria.informatik.gu.se

Open Source Movements as a Model for Organizing

Jan Ljungberg
Viktoria Institute, Göteborg, Sweden
Janl@viktoria.informatik.gu.se

***Abstract*- Open source software such as the operating system Linux has in a few years created much attention as an alternative way to develop and distribute software. Open source is to let anyone have access to the source code, so that they can modify it. Open source could be seen as a movement, where communities of highly skilled programmers collectively develop software, often of a quality that outperforms commercial proprietary software. These movements are based on virtual networking on Internet and the web. They are loosely coupled communities kept together by strong common values related to hacker culture. Work seems to be totally distributed, delegated and loosely coupled. The highly skilled members contribute by pride to the collective effort of free software development. In this paper the open source phenomena is investigated from different perspectives. In this paper it is claimed that the open source movements is one key to the understanding of future forms of organizations, knowledge work and business.**

I. INTRODUCTION

Open source software such as the operating system Linux has in a few years created much attention as an alternative way to develop and distribute software. It seems to be a successful way to create high quality software with little cost. But what kind of phenomena are open source movements? They are not virtual organizations in the sense described in the literature [7, 10, 11, 15]. These are mainly focused on networks and coalitions between companies or company units, not between individuals. Open source is also different from many forms of virtual communities, such as communities of practice [5], on-line communities [29] and virtual communities of more commercial nature [14].

An open source project is a loosely coupled community kept together by strong common values such as that software should be free. Work is totally delegated, but still kept together by one or a few coordinators. Its highly skilled members contribute with pride to the collective effort of free software development. Communication media as email and the web is of fundamental importance. Open source projects has been likened with a bazaar [27], i.e. a marketplace where people enter and leave, sell, buy and exchange goods. In this paper we will continue to refer to open source as a bazaar.

> "No quiet, reverent cathedral-building here – rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (…) out of which a coherent and stable system seemingly emerge only by a succession of miracles." [27].

One of the most well known examples is the operating system Linux. The development of Linux is totally distributed, anyone could download the code, contribute to it, send it back, and if it is good enough he will be in the Linux-developers Hall of Fame. The contributors to Linux come from all over the world, most of them have never met face to face. The contributors and other dedicated Linux-proponents interact through email and the web.

> "Within three years, this loose, informal group, working without managers and connected mainly through the Internet, had turned Linux into one of the best versions of Linux ever created." [21]

Communities of free software development are certainly not new, they have been around for a long time: related to specific hardware, operating systems, programming languages etc. (e.g. software to the Amiga-computer, the Apache web-server, Emacs, Lisp, GNU, Prolog, etc.). What really makes Linux special is the impact it has had on the commercial world. Around the non-commercial development, commercial companies are now making business on the open development of the Linux source, i.e. a non-commercial product development that threatens to take market shares from Microsoft.

Why are the open source movements interesting to IS-research in the first place? I believe that they contain several keys to the future of knowledge organizations and knowledge business. To reveal these keys several important questions has to be addressed: Can we find similar examples outside the software community and hacker culture? Is it entirely linked to development of intangible assets? Can we find bazaars inside large companies? Is the development of Linux so loosely coupled and distributed as it seems at first sight? Can we even talk about a bazaar mode or bazaar style of organizing? If so what can companies learn from the bazaar style? What parts of the bazaar could one apply?

In this paper the open-source phenomenon will be discussed from different angles. The aim is to problematize the phenomena and set out a direction for further research. First the history and basic characteristics will be outlined. Then several perspectives will be discussed: open source as knowledge sharing, as software process, as community based development, as customer relations, as organization of knowledge work, and as a business model.

BACKGROUND

A. *The roots of open source*

The roots of the free software and open source phenomenon is to be found in the hacker culture from the early sixties (see e.g. [20]). The communities related to the operating system Unix and the C programming language played an important role. When powerful Sun workstations running on Unix were introduced in the mid-eighties one big

challenge was to provide them with graphical user interfaces (GUIs). Among several proprietary GUIs (one offered by Sun itself), the X Windows System was taking the lead. One critical factor in its success was that the developers of X gave the source code away for free in accordance with hacker ethics, and was able to distribute it on Internet. Besides the "Unix-culture" a plethora of technical cultures rose and died, due to the rapid technical change. However the Unix technical culture became the mainstream of hacker culture. The problem was that the Unix operating system still was an expensive proprietary system. Other problems were that even if a system was released as free software, anyone could make a proprietary modified version of it. The X Windows System was an example of that, released as free software from MIT, but soon adopted by various computer companies and bundled into their proprietary Unix versions.

The open architecture of Unix made it easy to develop software for it. This has probably been of utmost importance for the growth of the open source movements.

Some of the open source projects have delivered software that is generally considered to be more reliable and more technically well designed than its commercial equivalents. Examples are the free GNU compiler that was generally considered to be more effective than similar commercial compilers and the Apache web-server that is considered to be the market leader.

B. Free software

In 1984 one of the free software pioneers, Richard Stallman started to write a free UNIX system, GNU. He left his job at MIT so that they couldn't claim the rights to the software and interfere with his aim to distribute it as free software. Stallman later founded the Free Software Foundation (FSF) as a charity organization for free software development. Free software has nothing to do with the price, but with the rights [30].

- You have the freedom to run the program, for any purpose.
- You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code).
- You have the freedom to redistribute copies, either for free of charge or for a fee.
- You have the freedom to distribute modified versions of the program, so that the community can benefit from your improvements.

To achieve this freedom, distribution terms were needed that prevented the GNU software to be turned into proprietary software. This was achieved by introduction of the copyleft method. Copyleft uses copyright law, but turns it the other way around: instead of a means of privatizing software, it becomes a means of keeping software free. The idea is that anyone is given permission to run the program, copy the program, modify the program and redistribute modified versions, but not permission to add restrictions on your own. Copyleft was implemented in the GNU General Public License (GNU GPL).

C. Open source

The term open source was at some point raised as an alternative to free software. By making open source a trademark it would be possible to keep the control over the concept, so that it wouldn't meet the same fate as the term hacker which due to media and press almost got the meaning criminal [8].

Typically, an open-source project starts with a single programmer solving a small problem (e.g. a malfunctioning system affecting his own work) that later turns out to be significant, and then make the solution available to others. The origin to WWW was the work by Tim Berners-Lee to help high-energy physicists share their work. Larry Wall originally wrote the widely used script language Perl to solve some problems in systems administration. Richard Stallman is said to have started the GNU project because a vendor would no longer provide source code for a malfunctioning printer driver (examples from [25]).

At the core of open source movements lies a culture that encourages people to contribute and share, i.e. getting credit for good contributions is what brings status and influence.

Ownership is an important issue in open-source projects. In this context the owner (a person or a group) of a software project are those who have the exclusive right, recognized by the community at large, to re-distribute modified versions of the software [28]. According to Raymond [28] there is basically three ways to acquire ownership of an open-source project: to found it; to have it handed over by the former owner; or to volunteer to take over a dying project where the former owner has lost his interest.

The owner used to be leader of the development, practicing a benevolent dictatorship (though there are a few other models discussed below). As the owner attracts contributors, i.e. people that discover the software and want add to its development, he/she becomes more of a coordinator or project leader. According to the free software and copyleft agreement, you can redistribute modified versions. However starting a new track, i.e. spawning the original project into new versions is rare, and against the free software community norms. This means that Linux kernel is controlled, basically by one person who makes the important decisions. He may then have a group of people helping him to e.g. review source code developed by other contributors. If an ordinary contributor thinks that the development goes in the wrong direction and start an own subproject, that will be strictly against the rules of the community.

D. Ideology

There is a quite wide span in the open-source ideology. One dimension is the zealotry, i.e. if the open-source development is viewed as a means to an end of producing free software of high quality, or if it is viewed as an end in it self [27]. At one end of the spectra there is great zeal and the view that open-source is a philosophy and way of life. At the other end there is no big deal about the ideology, rather than an opinion that open-source is a good way to build software.

Another dimension is the degree of hostility to commercial software. A very anticommercial person in the open-source

community might think that commercial software is theft and hoarding, while a non-anticommercial person might think commercial software is generally OK and may co-exist with open source movements [27]. In the first case writing free software could be seen as a resistance movement against large software corporations. Especially Microsoft plays an important role as the evil enemy. Historically, large parts of the hacker culture have been very zealous as well as very anticommercial, an example of this is the Free Software Foundation (FSF). Recently however, several large companies have joined open-source communities, playing the game basically on the same terms and conditions as any others involved. One example is when IBM joined the Apache Group on Apache's own terms, by contributing code fixes and features back to the open-source base, as well as contributing with other resources [13]. The development of Linux may also represent a more market-friendly strain.

An illustration of the clash between the open-source culture and the commercial terms of public offerings was when Red Hat was being filed for public stock offering [1]. Several people within the open-source community got special invitations to participate in the initial public offering. People from the community that made Red Hats existence possible in the first place, now had the chance to make some commercial benefits. In the fine print on the special invitation, it was stated that people had to be approved by the company handling the stock distribution, based on investment experience and financial background. Since these criteria are not qualities at the heart of the open-source community, it was not uncommon that hackers that been involved in coding Linux were denied approval to buy stock. This generated strong reactions on Slashdot and similar Linux forums.

> "They're auctioning my software off on the New York Stock Exchange to the highest bidder, and I can't take part. We coders had been abruptly disenfranchised, after having had silver carrots waved in front of our noses. I'd opened my first money-market account just now, in order to take part in the commercial future of something I believed in -- and the door had been slammed in my face." [1].

An interesting issue is what will happen to the open source communities, and their values when their work ends up at the stock exchange. Compare this to the current trend in knowledge intensive companies, offering stock to the employees for free or at favorable prizes.

E. Meeting places

Internet and email had been used by the free software communities (as well as by many computer scientists) for quite many years before the public discovery of the Internet. The development of Linux coincided with a more public use of Internet and the explosion of the web in the 1990s. This meant a lot for free software communities, especially when Linux got a wider public interest. With meeting places on the web the Linux-community grows and is active at several places on the web. One of the most well known is Slashdot (www.slashdot.org), a vertical web-portal. A vertical web-portal has a narrow focus, i.e. at Slashdot only Linux issues are discussed. The site functions as an interactive forum for discussions as well as library, marketplace, coordination medium etc. for Linux enthusiasts. The site is run like an open source project, there is one basic owner/editor with some people helping him, and lots and lots of people contributing by discussions, articles and advice.

Slashdot has so many hits every day that a phenomenon has occurred that is called the Slashdot effect. If you write an interesting paper on Linux and it is published on Slashdot, the paper will physically remain on your own server and Slashdot will link to it. The consequence of this is that your web-server will have so many unexpected hits that your web-server will shutdown. There is even a meta-slashdot effect, i.e. if you write an interesting paper on the Slashdot-effect, your web server may be shutdown of all the hits. This highly active and well visited web-site has had a great impact on decisions in several companies. Debates and appeals among community members has put pressure on vendors strategies for aligning products with the Linux operating systems.

F. Examples of Open Source Projects

The GNU Unix project started 1984, when the legendary hacker Richard Stallman, left MIT to write GNU software. The name GNU follows a hacker tradition, being a recursive acronym for *GNU is Not Unix*. Stallman used and adapted existing pieces of free software, and in some years the GNU project came to develop a set of tools and programs running under Unix, e.g. a GNU compiler and the editor GNU Emacs.

The Apache web-server project started in early 1995, due to the reason that the most popular public domain web server software stalled after the developer left. A group of web masters gathered via private email to share and coordinate their efforts in changing and improving the server. One person volunteered the use of his server as a shared information space for the eight core developers (located in U.S., Canada, and Europe), and a public mailing list for communication [13]. Later on IBM joined the community on Apache's terms, contributing to the open source base. Apache web-server is now generally considered to be the market leader in its domain.

Linux originally started as a project for writing a Unix system for a specific PC processor architecture (Intel x86). Today it is the most widely ported operating system available for PCs. The project was initiated by a student in Helsinki (Linus Thorvalds), who started to write the kernel of the operating system in 1991. He gathered a few supporters who appreciated getting and solving fixes. The OS-kernel quickly came to support the functionality expected of a Unix-kernel. There where other free Unix systems for home computers competing at the time, e.g. 386BSD, but while 386BSD is a version of Unix, Linux is a Unix like system with the kernel written from scratch. At the same time different tools from the GNU project were becoming well established in the Unix community. The fitting of these applications, as well as other open-source products, to the Linux kernel was part of the success [3]. Commercial businesses like Red Hat and Caldera then grow around Linux, with their main business to package versions of free software to be more user friendly and easy to install, together with manuals, education and support. From

one persons attempt to solve a problem, via a community of contributors, Linux is now threatening to take shares from Microsoft. Some computer sales firms now sell PCs pre-loaded with Linux instead of Microsoft Windows.

Mozilla is Netscape's version of the Communicator suite that in 1998 was announced to be open source. Inspired by Eric Raymond's now classical paper "The Cathedral and the Bazaar" [27] Netscape decided to give away both the browser and the source code for the whole suite, except parts that where protected by other patents or copyrights [15]. To invite people to contribute, to make source code, bug reports and coding instructions available, Netscape started the web-site www.mozilla.org. Mozilla is thus a release of a formerly proprietary product from a well-established commercial company.

II. Key Dimensions of Open Source Movements

A. *Gift Economy versus Scientific Knowledge Sharing*

The open-source communities are often analyzed as a form of gift economy or gift culture [27, 19]. The fundamentals of gift economies are the obligation to give, the obligation to receive and the obligation to make a return for gifts received [22]. A gift transaction thus involves a usually unstated obligation to repay the gift at some future time without any bargaining or demands that the gift should be repaid. [19]. In a gift culture social status is determined not by what you own or control but by what you give away. The giving of gifts is therefor a way to power and control.

The contrast to a gift exchange is a commodity transaction. While social relations drive gift economies, commodity economies are driven by price. The former benefits from improving the "technology of social relations", the latter benefits from improving the technology of production.

Gift giving as classically defined certainly is common on Internet, e.g. peers emailing each other useful information. However much of the help and sharing that goes on is actually different than traditional gift exchange [19]. When people on Internet share useful information, the recipient is often unknown to them, and might not even be part of the same community. Gifts of information and advice are often given to groups or communities as a whole, rather than individuals. Another difference to traditional gift exchange is that after giving away source code, information or knowledge, the gift is still in possession of the giver. It is an infinite resource.

Open source movements such as gift economy is thus rather an example of public good, i.e. the software is made available for free to anyone, regardless if they have contributed to the project or not [19]. There is a serious temptation to free-ride in such a project, i.e. to let other people write the program and then enjoy the fruits of their labor, thus there is a risk that the project will fail if not enough people volunteer.

In an open-source community the only available measure of competitive success is reputation among one's peers. That reputation is gained by giving away or sharing with others high quality software, knowledge or solutions to problems. Good reputation is a primary reward in itself, but it is also a way to attract attention from others. This attention may eventually give you credit also outside the gift economy of the open-source culture, e.g. in terms of status, job opportunities or money.

The role of gift economy in modern society is not well researched, and a barrier to this end seems to be a tendency to see them as archaic customs [22, 9]. Gift economy seems to be the opposite of market relations, but the way it works out in many contexts is not so different. Sociologists often describe gift giving as a process of exchange where individuals rationally pursue their self interest, i.e. giving to others is only motivated by the expectation of some reward, direct as power over others, or indirect as social approval.

Academic research has much resemblance of the gift economy, in the sense of non-altruistic gift transactions. In the academy you give away your knowledge not because you are good, but because that's the way you do career within this community. You give away knowledge and information in return for status and reputation. By a system of peer review your contributions are judged if they are good enough to enter the field. Peer review is a social mechanism through which a discipline's experts or the core members of a community maintain control over new knowledge entering the field [23; 11]. With the editorial referee system for journal publications as the archetype, peer review is by now a fairly established academic designation for marking the scientific quality of a piece of work, for assigning a candidate to a position or for approving or rejecting a research proposal. All established researchers are more or less involved in peer review processes, as reviewers of books, articles and project proposals or as being reviewed themselves.

By sharing knowledge and being open about results and methods, results can be justified and replicated. Others can give contributions by responding, or by continuing on the published work, pushing the scientific frontier. By writing and publishing papers and by being referred to by others, you not only share your knowledge, you become visible in the academic community. By doing something useful that others can benefit from, and share it with them, they give credit by referring.

The open source communities are driven by the same norms, hardly surprising because they stem from early academic computer science communities. You write a piece of software and afford it to the community. Your contribution is peer reviewed, and if it is good enough you get your credits in the open source/academic gift economy/market. You trade your contributions in return of reciprocal contributions that you can use in your research, but most importantly for credits. In the attention economy you can trade these credits for money.

B. *Software Process*

A lot has been written on improving the software process, project coordination, software process maturity etc. Since the bazaar mode seem to be best (or only) applicable to the development of already existing code [27], e.g. testing, debugging and improving, many of these writings may not be relevant. Let us anyhow try to relate the bazaar mode to some classical papers in software process improvement.

Humphrey [17] introduced five levels of software process maturity, to judge how good the performance in an organizations software development is:

- Initial – There is no statistical control and thus no orderly process improvement is possible.
- Repeatable – The organization has reached a stable process by rigorous project management of commitments, cost, schedule, and changes.
- Defined – The process is defined and consistent, advanced technology can now be used.
- Managed – Initiated process measures beyond cost and schedule performance has been introduced.
- Optimized – There is a foundation in place for continued improvement.

For now, we will only discuss the first two levels, they will be the most applicable to discuss the bazaar mode. To reach the second stage, four basic project controls have to be implemented:

- Project management's fundamental role is to ensure effective control of commitments.
- Management oversight by review and approval of all major development plans before official commitment.
- Quality assurance by controlling that the software development work is done the way it is supposed to be done.
- Change control by introducing changes in a managed an orderly way.

Let us try to look at the open source way of development from these four requirements. All above issues relate to different aspects of controlling commitments, design, quality and change.

One of the most common models in coordinating open-source projects is that several contributors work under a single *benevolent dictator* who owns the project. Typically this organization evolves when the founder attracts contributors. The benevolent-dictator has the right to take decisions (e.g. of what to include for redistribution) and has an obligation to credit contributors fairly. By contributing to the project you earn part of its reputation in return. When a project grows it is common that it will be split in subsystems with subsystem-owners, i.e. introducing two kinds of contributors: ordinary contributors and co-developers.

Controlling commitments will probably not be relevant, since people volunteer by interest and for free, controlling them would rather have a negative effect. They are not enrolled on any other commitment basis than their own, and their will to contribute. Their reward is status in the community. They will in most cases also be many, so many that from parallel solutions the best one will be chosen, or the different solutions will form a synthesis, taking the best parts from several solutions [27].

Development, design and planning seem to be totally decided on by the owner eventually in cooperation with the co-developers, and with suggestions from the ordinary contributors. Thus, the management oversight seems to be inherent in the model.

An important issue is whether the coordinator of a bazaar mode project needs to be a talented designer. According to Raymond [27] it is absolutely critical that he/she is able to recognize good design ideas from bad ones. That is, even though he might not come up with all design ideas himself, he must be able to judge them. Looking at the track record of open source projects there is reason to believe that the project leader/dictator use to be a good designer. The stories of the open source projects are full of descriptions of the importance of the founders as entrepreneurs, as dedicated persons, as technical innovators, and good designers. Brooks [4] claim that great design comes from great designers, may thus be valid in the bazaar mode as well. Studies has showed that what is common to good designs and successful software projects is that they were led by great designers, i.e. great designers mean more for the success of a software project than which method that are used.

Quality assurance seems to be at the hart of the bazaar mode. An army of testers, debuggers, and programmers contribute to the development. The leader and his co-developers review the contributions and choose the ones that hold the best quality.

The last point in managing change in an orderly way, may be the most obviously fulfilled criteria. Since the dictator has the ultimate power, he will take the decisions of what to change, what source code to implement, and what bug fixes to include. As Linus Thorvalds state it:

> “So even though a large number of people work on Linux, the core kernel remains something that I can keep track of.” [32, p.39]

An alternative to one single benevolent dictator is *rotating dictatorship*, e.g. used in the development of Perl. Yet an alternative is to form a *voting committee* with the co-developers. This has been used in some large projects such as the Apache web-server. The project is managed by the Apache-group, a geographically distributed group of volunteers using Internet and the web to develop and distribute the server software. Since there is no CEO to take decisions, a system of voting via email based on minimal quorum consensus is used [13].

According to Raymond [27] one cannot start from scratch to have a successful bazaar mode project. One can test, debug and improve, but hardly originate in bazaar mode. All successful bazaar projects have been based on already available software. In some senses a bazaar is very similar to an ordinary software project and in some senses not. There is often a talented designer acting as a strong project leader, as in many successful software projects. However, there are also lots of skilled volunteers working for free, forming a community with shared values.

Brooks [4] argued that software development suffers from several essential difficulties inherent in software development: complexity, conformity, changeability, and invisibility. These essential difficulties he argued, will always be there, and there are no magical tricks in form of methods or tools that will once and for all solve them, i.e. no silver bullet to kill beast. Bazaar style of software engineering is best suited for improving on existing system, and the

essential difficulties of software development remain. The bazaar is not a silver bullet either.

C. *Participatory User Driven Design*

The operating system Unix is well known to be difficult to master. The mastery of Unix is part of the culture, e.g. the skills and tricks, the knowing of all acronyms in the command language, the fame of being a super user. The users of Linux and other open source software used to be technicians, constituting a community where developers and users in many cases were the same people. This community is now being broadened, with more diversified members.

Can companies create better products by involving its customers by supporting customer communities and invite them in product development, i.e. can the bazaar mode reach out to ordinary end users? One example of this is to provide access to the source not to the whole world, but to the paying customers' [24]. By using web technology it is possible to involve customers or potential customers in a dialogue of product improvements and future design. This would be possible in other cases than software development. Ultimately this will lead to a in power shift to the customers benefit.

D. *Virtual Organizing*

Another angle to discuss the bazaar from is to relate to the discourse on virtual organizations. There is no general agreement on what a virtual organization is. Some examples of what could be found under the virtual/network umbrella are: virtual corporation [10]; network organization; virtual organization; imaginary organization; network enterprise etc. [6, 7, 10, 12, 16]. Capturing most of the essence of virtual organizations, Byrne [6] describes it as:

> "[…] a temporary network of independent companies – suppliers, customers, even erstwhile rivals – linked by information technology to share skills, costs, and access to one another's markets." [6]

The most research on virtual organizations is focused on different kinds of networks and coalitions between organizations, not between individuals. Even if companies join a bazaar, the collaboration will be between individuals. It is individuals that perform work, not institutions. Neither, does the mentioned research include the idealistic or non-commercial dimension of the bazaar organization (the term non-commercial may be a bit inappropriate, since you could actually sell free software, but the eventual commercial relations among the involved stakeholders don't seem to be well regulated). Let us take some views on virtual organizations and relate them to bazaar organizing.

Introna [18] makes four main points of virtual organizations:

- An enterprise that can marshal more resources than it currently has on its own, using collaborations outside its boundary.
- The use of technology for a wide array of strategic alliances to grasp specific market opportunities.
- A collection of management theories
- A network or loose coalition of manufacturing or services uniting for a specific business purpose.

What then is a bazaar compared to these points? It could be an enterprise (e.g. Netscape in the case of Mozilla) but most often it has been an individual entrepreneur, or a team that marshal more resources than its own, using collaboration from other people or companies, basically on a volunteer basis. It uses and is heavily reliable on technology and infrastructure as email, Internet and the web, but not really to grasp specific market opportunities. Rather the driving forces are for homesteading the technological wasteland, i.e. to fill technical or functional gaps. This may be equivalent to market opportunities, but from a dedicated technicians point of view, not from a management or economic point of view. Grasping for market opportunities are commercial, homesteading is idealistic. It is not a management theory (yet), and it is not a coalition of manufacturing or service uniting for a specific business purpose, at least not in the usual meaning of business purpose.

To avoid infotechnical overtones of the term virtual, Hedberg et al. [16] prefers the term imaginary organizations:

> "Imaginary organizations are organizations where important processes, actors and resources appear both inside and outside of the legal unit of enterprise, both outside and inside of the accounting system and of the organization charts. Markets and hierarchies are interconnected through networks of cooperating people and coordinating information technology." (Ibid. page 2)

In bazaar mode actors as well as resources appears both inside and outside any kind of unit that one could think of. In contrast to how we normally think of companies, even if they are networked, the relations in the bazaar are in many ways highly unregulated in any legal sense. The law has been replaced by trust. When technology development is as fast as it is, the value of owning the technology may decrease. When the value is created around the technology in the form of services and support, branding will become more important than patents.

It could be seen as a mixture between hierarchy and market. There is often a strict hierarchy with a team or a benevolent dictator at the top, eventually co-developers next to the top, and then the community of contributors. It is also a market in the sense that the best contributions win, and as an individual you get credit and reputation by contributing. The bazaar is certainly a network of people using information technology to interact and to coordinate. Most of all it seems to be a mixture of several forms. It has a goal and a hierarchical structure, at the same time it is loosely coupled in the form of a community regulated by norms rather than rules.

E. *Business Model*

Commercial businesses tied to open-source projects provide additional resources for developing the free components of the software, but even more importantly it helps to promote the open-source packages and drive them

into the mainstream [26]. Companies make money out of open-source software in at least three ways: on distributing the open-source software; by adding value to the open-source software by additional proprietary products; and by relating to open-source software in different ways, such as bundling it with own products [24].

The first way to make money on open-source is to package the free software in a user-friendly way, and then sell it together with books, manuals, training and support under a trusted brand name. Red Hat Software is the best example of this business model [33]. Red Hat shrink-wraps software in different packages, including the Linux operating system and tool sets, under the GPL license (see above). They also offer different support packages. It is the services and support that is key to Red Hat's business not product sales. Under this branding and distribution model, customers pay for three things: the source code on a CD-ROM; a company's stamp of approval on the code as the (at the moment) latest and most stable version: and a corporate commitment to support the distributed software. Another example of company in this category is Caldera (www.caldera.com).

As open-source packages becomes popular and gets a corporate use, a gap develops between the needs of the user community and the features provided by the core developers. The user community demands additional services and are ready to pay, which generates new business opportunities [26]. To link the world of open-source developers and commercial customers a hybrid business model have been used.

The hybrid business model for open-source is to develop proprietary products that add value to the open-source software. These products are mainly sold to the commercial community, and since they are proprietary they are not contributed back to the open-source community. Sendmail, Inc. (www.sendmail.com) is one example of this model, providing the email delivery system Sendmail. The founder of the open-source program Sendmail needed more resources to develop the software. This together with the factor that proprietary mail formats undermined the open SMTP e-mail standards urged him to start a company. By starting Sendmail Inc. both the dominant market position of Sendmail could be preserved, and the threat of open e-mails standards could be met.

> “Sendmail, Inc. develops commercial products and services for ISPs and enterprises for whom email is mission critical, while continuing to drive innovation and standards through Open Source™ software development.” (Sendmail, Inc.)

The hybrid model brings value to both the open-source market and the commercial market. New functionality and standards initiatives appear first as open source. This leads to increased speed of innovation and quality improvements. Through inspection and use by a world-wide network of skilled technicians. Additional development resources for open source are funded, as well as for requirements that are unique to the commercial customer base. Other examples of the hybrid model are Scriptics supporting the Tcl scripting language and Cygnus supporting GNU tools [26]. Both sell extended versions for special commercial applications as well as they develop the open-source core and distribute it freely. Founded in 1989, Cygnus was the first company making business on open-source [31].

These companies fill in the gaps of the open-source model by providing products and services that otherwise would not have emerged under that model. By adding proprietary value to open-source, they make money by selling extensions, tools, hardware, support service, custom engineering services etc.

O'reilly [24] describes a third business model, which he calls "make your money on the side strategy". In this model the company's focus is not in open-source or value-adding proprietary software. These companies are ordinary vendors relating in different ways to open-source and taking more care of customers already relying on open-source software. Examples of this third model are IBM, Netscape and Oracle. IBM is shipping the Apache web-server with its own WebSphere Application Server. They will also provide commercial support for the Apache server. IBM hopes to gain credibility for its own web application servers, by shipping them with Apache, which has credibility as market leader. Another example is Netscape's release of Communicator's source code. The strategy is to accelerate development and distribution of future versions of the product. Oracle will, like several other companies, port its products to Linux, expanding the number of potential users of their products.

For ordinary vendors of non open-source products, aligning with open-source software may bring benefits in several ways:

- By providing good will
- By helping to reach a larger market
- By helping to develop the products

The open-source movement and commercial activities seems to be symbiotic. The commercial activity gains value from non-commercial product development and at the same time boost the interest for the product, giving value back to the community. By using commercial resources for developing free software the open-source community is helped and grows, which also increases the market for the additional products and packages for companies involved.

CONCLUSIONS

In this paper the characteristics of the bazaar model has been outlined, i.e. as it is expressed in open-source movements and the free software community. A number of phenomena that already have influenced or will influence organizations in the future have been suggested. These dimensions have to be further and deeper researched.

The bazaar is a strange mixture of different organizational forms. You share source code and knowledge, but you get credits back that you could eventually cash in later. It could be viewed as gift economy and scientific culture of sharing, and at the same time a market. It contains hierarchy, but at the same time it is a swarming marketplace. It has some similarities with ordinary software projects, but while they are based on teamwork, the bazaar is based on community. It

could be seen as a virtual organization, but where trust has replaced law in regulating relations, and where the network consists of individuals rather than institutions. It has generated several new business models, and is at the same time infused with anti-commercial ideological values.

The number of open-source software projects is growing, and the track record makes it a promising way to develop software, at least when it comes to test and improve. Companies that make money on open-source software are increasing in number and new business models are invented. In the meantime there are lots of questions to address: Is the bazaar mode only applicable to software development? Could work inside large organizations be organized as loosely coupled communities? How can companies in other business areas apply to the business models of open source?

I believe that the bazaar mode of open source projects will influence the future of knowledge organizations both in terms of organizing, customer relations and business models. Hence the most important influences from open source to learn from and to study further are:

- as a way of knowledge sharing;
- as a way of coordinating development projects;
- as a customer relationship model;
- as an organizational model;
- as a business model.

ACKNOWLEDGEMENTS

This work was funded by the Swedish National Board for Industrial and Technical Development (NUTEK), through the Competitive KIFs project within the AIS-program. I also would like to thank Magnus Bergquist, Jens Bergquist, Mathias Klang and Peter Ljungstrand at Viktoria institute and Anders Lundkvist at Stockholm School of Economics for valuable comments.

REFERENCES

[1] Ananian, C. S. (1999) A Linux Lament. July 30, 1999. <www.salon.com/tech/1999/07/30/redhat_shares/print.html> (10 Aug. 1999).

[2] Behlendorff, B. (1999) Open Source as a Business Strategy. In *Open Source: Voices from the Open Source Revolution*. O'Reilly & Associates. Also available at: <www.oreilly.com/catalog/opensources/-book/toc.html>. (28 July 1999).

[3] Bensson, R. (1995) The Humble Beginnings. *Linux Journal*, June 10, 1999. <http://Linuxjournal.com:8080/lj-issues/issue11/history.html>. (20 Aug. 1999).

[4] Brooks, F. (1987) No Silver Bullet: Essence and Accident of Software Engineering. *Computer*, Vol. 20, No. 4 (April 1987).

[5] Brown, J. S. and Duguid, P. (1991) Organizational Learning and Communities of Practice. *Organization Science*, Vol. 2, No. 1, p. 40-57.

[6] Byrne, B. (1993) The virtual corporation. *Business week*, 8 February, 1993.

[7] Castells, M. (1996) T*he rise of the network society*. Padstow, Cornwall: Blackwell Publishers.

[8] Chalmers, R. (1999) The original upstart. *Linux World*. <www.softpanorama.org/OSS/postulates.html>. (15 July 1999).

[9] Cheal, D. (1988) *The Gift Economy*. London: Routledge.

[10] Chubin, D. E. & Hackett, E. J. 1990: *Peerless Science. Peer review and U.S. Science Policy*. Albany: State university of New York Press.

[11] Davidov and Malone, (1992) T*he Virtual Corporation*. New York: Harper Business.

[12] Drucker, P. (1988) The coming of the new organization. *Harward Business Review*, 1988, January-February.

[13] Fielding, R. T. (1999). Shared Leadership in the Apache Project. *Communications of the ACM*, Vol. 42, No. 4, 38-39.

[14] Hagel III, J. and Armstrong, A. G. (1997) *Net Gain*. Boston: Harvard Business School Press.

[15] Hamerly, J., Paquin, T. and Walton, S. (1999) Freing the Source the Story of Mozilla. In *Open Source: Voices from the Open Source Revolution*. O'Reilly & Associates. Also available at: www.oreilly.com/catalog/opensources/book/toc.html. (28 July 1999).

[16] Hedberg, B., Dahlgren, G,. Hansson, J.and Olve, N. (1997) *Virtual Organizations and Beyond. Discover Imaginary Systems*. London:Wiley, 1997.

[17] Humphrey, W. S. (1988) Characterizing the Software Process: A Maturity Framework. *IEEE Software*, Vol. 5, No. 2 (March 1988).

[18] Introna, L. (1997) Thinking about virtual organizations and the future. In *Proceedings of ECIS'97*.

[19] Kollock, P. (1998) The Economies of Online Cooperation. In M. Smith and P. Kollock (eds). *Communities in Cyberspace*. London: Routledge.

[20] Levy, S. (1984) *Hackers*. Middlesex: Penguin Books.

[21] Malone, T. (1998) The E-Lance Economy. *Harward Business Review*, September-October, 1998.

[22] Mauss, M. (1950/1999) *The Gift*. London: Routledge.

[23] Merton, R. K. and Zuckerman, H. (1973) Institutionalized Patterns of Evaluation in Science, in Robert K. Merton, *The Sociology of Science*. edited by Norman W. Storer. Chicago:The University of Chicago Press.

[24] O'Reilly, T. (1998) *The Open Source Revolution*. Release 1.0, Esther Dyson's Monthly report. <www.edventure.com/release1/1198.html>. (22 Aug. 1999).

[25] O'Reilly, T. (1999) Lessons from Open Source Development. *Communications of the ACM*, Vol. 42, No. 4, 33-37.

[26] Osterhout, J. (1999). Free Software Needs Profit. C*ommunications of the ACM*, Vol. 42, No. 4, 38-39.

[27] Raymond, E., S. (1998a) The Cathedral and the Bazaar. www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html. (15 June 1999). Also in Raymond, E, S: T*he Cathedral and the Bazaar*. Sebastopol: O'Reilly, 1999.

[28] Raymond, E. S. (1998b) Homesteading the Noosphere. www.tuxedo.org/~esr/writings/homesteading/homesteading.html. (15 June 1999). Also in Raymond, E, S: *The Cathedral and the Bazaar*. Sebastopol: O'Reilly, 1999.

[29] Rheingold, H. (1994) *Virtual Community*. London: Minerva.

[30] Stallman, R. (1999) The GNU Operating System and Free Software Development. In *Open Source: Voices from the Open Source Revolution*. O'Reilly & Associates. Also available at: www.oreilly.com/catalog/opensources/book/toc.html. (28 July 1999).

[31] Tiemann, M. (1999) Future of Cygnus Solutions: An Entrepreneurs Account. In *Open Source: Voices from the Open Source Revolution.* O'Reilly & Associates.

[32] Torvalds, L. (1999). The Linux Edge. Communications of the ACM, Vol. 42, No. 4, 38-39.

[33] Young, R. (1999) Giving it Away: How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry. In *Open Source: Voices from the Open Source Revolution*. O'Reilly & Associates. Also available at: www.oreilly.com/catalog/opensources/book/toc.html. (28 July 1999).