

**Center for Activity Theory and  
Developmental Work Research**

Juha Siltala,  
Stephanie  
Freeman and  
Reijo Miettinen

# **Exploring the Tensions Between Volunteers and Firms in Hybrid Projects**

Working Papers 36 / 2007



# **Exploring the Tensions Between Volunteers and Firms in Hybrid Projects**

Juha Siltala  
juha@siltala.net

Stephanie Freeman  
stephanie.freeman@helsinki.fi

Reijo Miettinen  
reijo.miettinen@helsinki.fi



UNIVERSITY OF HELSINKI  
CENTER FOR ACTIVITY THEORY AND  
DEVELOPMENTAL WORK RESEARCH



## Contents

Abstract .....	5
Introduction.....	5
The Constitutive Features of the Open Development Model .....	8
GNOME and OpenOffice.org as Hybrid FOSS Organizations .....	11
The Desktop Integration Bounty Hunt: Who Decides What Features Are Important? .....	14
The failure of the Groupware project: Sun closes the code .....	19
Conclusions .....	23
References .....	27



## Abstract

Software development has traditionally taken two separate forms of organization, open source and proprietary. In recent years, a new hybrid model has emerged as an attempt to combine the benefits of the open development model with the hierarchical control over the development of commercial software products. In this paper, we define the constitutive features of the open development model and examine their realization in two hybrid software projects with different origins. We support the reconstruction of the concept of a hybrid as a dynamic interface between two co-existing models instead of a new model of software production.

**Keywords:** Free and open source software, hybrid projects, open development model

## Introduction

In this paper, we focus on tensions between the goals and ways of working of volunteer communities and sponsoring companies in two hybrid Free and Open Source Software (FOSS) projects. Such projects have been called *hybrids* because they attempt to combine elements of two, previously separate traditions of software production, open source and proprietary. These hybrids have strong corporate backing, which tends to bring the logic of profit-making and the culture of hierarchical control into the production of code. The sponsoring companies are trying to exploit the benefits of an open development model, which is based on open access to the code and invites any user to participate in the development of the code through the Internet.

In early phases of computing, the computer was more important than the software. Machines were expensive and rare. Customers who acquired them often wrote the software themselves for the specific purposes the machine was used for. The importance of software increased after computers became more affordable and commonplace, and after networking and interactive computing called for general-purpose operating systems and programming languages. The rise of software resulted in two lines of software design and distribution in the 1980s. Commercial software firms, such as Microsoft, started developing and marketing proprietary general purpose software for the end-user market. They

sold binary copies of their software while hiding the program source code behind restrictive licensing, trade secrets and patents. Meanwhile, research institutes and universities shared their software creations and improvements under permissive licenses that did not restrict the use and copying of the software for any purpose. This practice was institutionalized in the establishment of the GNU project's General Public License (or "copyleft") in 1989. It decreed that all users of the program have the right to use it, copy it, modify it and distribute their modifications. The GPL forbids the privatization of deviations from software distributed under copyleft. Derived versions also must be distributed under copyleft license. The most prominent example of this model of licensing and open development model during the 1990s was the development of the Linux operating system kernel.<sup>1</sup>

It is largely recognized that the open development model is a new phenomenon and a new form of work made possible by the development of the Internet. It has been defined as a new mode of software production (Dalle et al. 2004) and a new software business model (McKelvey 2001). Most of the analysts, however, give to the open model a meaning beyond software production. It is seen as a paradigmatic case of a network organization called for by the knowledge-based economy (Castells 2000) or information technology revolution (Freeman and Louçã 2002). Since it can be used in addition to software in the design of other types of products as well, it can be regarded a model of an Internet-based form of work (von Hippel 2005) or distributed work in and across organizations (Moon and Sproull 2002), a community-based model of knowledge creation (Lee and Cole 2003) and more generally a model of distributed innovation (Sawnehney and Pandelli 2000, von Hippel 2005) or knowledge creation similar to open science and culture (Boyle 2003).

What is the future of the model and what will be its relationship to firm-based closed models of software production is, however, contested. As far as the last question, a significant feature of the development of the 2000s has been attempts to combine or mix the two models. The first attempts were made in the late 1990s. Netscape freed its Web browser source code in 1998 and publicly released it on the Internet, and the subsequent formation of the Mozilla.org FOSS project resulted (Hamerly et al. 1999). This *top-down* model of hybrid project formation was adopted later by Sun when it established the OpenOffice.org office suite project.<sup>2</sup> Another *bottom-up* manner of utilizing open development in commercial efforts is to offer corporate backing and guidance for an existing FOSS community. An example of this is the GNOME user

---

<sup>1</sup> For an overview of the different "property regimes" in software development, see de Laat's (2005) analysis.

<sup>2</sup> Project homepage: <http://www.openoffice.org>. OpenOffice.org is a free open source office suite similar to Microsoft Office.

interface project,<sup>3</sup> which originated as a group of volunteer programmers. A number of companies later started to sponsor open projects, to take part in their development efforts, and to utilize their results in commercial products.<sup>4</sup>

Recent literature supplies different visions of how the development and relationship between the two models will evolve. In her study on these competing software business models, Maureen McKelvey (2001, 233) evaluates the potential of open development or 'network-based models' to replace firm-based control in software production. She concluded (*ibid.*): "The analysis shows, however, that this network-based forms of organizing creation of novelty is more a complement than a substitute for firm-based innovation." She further suggests that over time, "both firm-based and network-based models seem to change towards the hybrid model of network development of knowledge, with a firm-based appropriation of economic returns." Von Hippel and von Krogh (2003, 209) also think that open source development can be regarded as a private-collective model of innovation that can offer society the "best of the both worlds" under many conditions.

In his study of open source software, Steven Weber (2004) in turn suggests that both open and firm-based models will coexist over time as distinct historically developed models of production. He thinks that the interface between the differently structured systems is the most creative place "where new forms of order and organization arise" and that "this is also the place where the relationship between the open source process and more traditional forms of organization are worked out" (2004, 262). The key theoretical challenge for him is "to conceptualize more clearly how hierarchically structured organizations (like large governments and corporations) develop and manage their relationships with network organizations." Finally, Eric von Hippel (2005) has recently presented a radical view of the future of innovative activity, in which firms will externalize the development of new products and services to "innovation communities" or networks of users mediated by the Internet and computer-based shared means, such as Computer-Assisted Design (CAD). The firms' strategies will be based on the utilization of the ideas and prototypes created by these communities, and firms will not own those ideas.

Following Weber (2004), we think that the hybrids are the place or interface where the future of FOSS and its relationship to proprietary and hierarchical forms of organizations are worked out. Corporations and FOSS

---

<sup>3</sup> The GNU Network Object Model Environment (GNOME) is part of the GNU project's (<http://gnu.org>) effort to build a completely free computing environment modeled after the Unix operating system. GNOME is the GNU project's graphical user interface, or "desktop environment."

<sup>4</sup> As examples of different kinds of large and complex open source software projects, OpenOffice.org and GNOME are objects of our ongoing studies. Empirical data for this paper is extracted from them.

projects have their own specific sets of historically developed tools, rules and norms. When a combination of both is attempted in a single project, many kinds of tensions and conflicts are likely to emerge. The vitality of a hybrid project depends on how these tensions and contradictions are regulated. On the other hand, hybrids are also important in defining how the emerging relationships between the models should be understood and defined.

In this paper, we will study two hybrid corporate/volunteer FOSS projects, GNOME and OpenOffice.org. A critical event from both projects and discussions related to them are analyzed to clarify the problems faced when the two models are combined. These events also reveal the types of tension between the volunteer communities and firms. Finally, we will draw conclusions from these cases concerning the future of the open development model and its relationship to firm-based software production.

## **The Constitutive Features of the Open Development Model**

To deal with the problem of compatibility and the contradictions between open and proprietary forms of software development, a definition of the specific nature of the open development model is needed. Several authors have developed definitions of the features or principles of the model (Raymond 1999, Feller and Fitzgerald 2002, Weber 2004). We think that it is sensible to define its constitutive features and distinguish them from other, contingent features often attributed to the model. We suggest that the two constitutive features of the open development model are an open code and the extensive distributed involvement of a large number of volunteers in the production of code. The contingent features are related to the various motives of the volunteers, their ideological commitments as well as the organization of work and type of licenses related to the projects. The constitutive basis of the open development model is that the code is freely available. Being a shared object of work for many, its unlimited availability is a basic condition for a distributed way of working. Accordingly, attempts to close or hide the code can be seen as contradicting the model.

The open development model is said to offer advantages over closed, in-house development as a model of organizing development work (Moon and Sproull 2002). These advantages relate to the quantity and heterogeneity of the programmers and users involved in development. The maxim "Given enough eyeballs, all bugs are shallow," dubbed "Linus's Law" by Eric Raymond (1999, 41), refers to a quantitative explanation. Raymond also presents a complementary qualitative explanation, the utilization of localized variety or the Delphi effect (Raymond 1999, 43): "Because adding more users adds more

different ways of stressing the program. (...) Each one approaches the task of bug characterization with a slightly different perceptual set and analytical toolkit, a different angle on the problem." The variety of skills, uses of software and the working environments of the volunteers, in addition to their sheer number, adds extra value to the quality of code (von Hippel 2003). In their study of Apache security software, Franke and von Hippel (2003) see the heterogeneity of user needs leading to a modification of the software for local purposes. Lessig summarizes the principle of the political economy of FOSS in a way that combines quantity and the heterogeneous motives and capabilities of the contributors (Lessig 2005, 24):

Under the conditions of anti-rivalness, as the size of the internet-connected group increases, and there is a heterogeneous distribution of motivations with people who have a high level of interest and some resources to invest, then a large group is more likely, all things being equal, to provide a good than a small group.

FOSS has been characterized by the particular motivation and worldview of the contributors. It has been characterized as a "hacker ethic" (Himanen 2001, Levy 1994) or, as Linus Torvalds suggests, acting "just for fun" or for personal satisfaction (Torvalds & Diamond 2001). Another version of the "hacker ethic" type of definition is that hackers are committed to the freedom of information (and freedom in society more generally) as depicted by the GNU project, Richard Stallman and the term "Free Software." An alternative account of motivation is based on the use value of the program both for the contributors and for users as a central incentive for contribution. In his classic collection of essays on programming work, Brooks (1982, 7) suggested that the usefulness of one's work to others is one of the central "joys of the craft." According to Weber, a distinctive feature of open source development is that "the product is perceived as important for a critical mass of users" (2004, 271).

We think that a particular motivation or worldview is a more contingent feature of the open development model than the two defined above, open source and distributed variety. The "ethic" account tends to romanticize the early generation of code-makers. As for the ideal of freedom, some recent surveys (International 2002, Stanford 2003, Shimizu et al. 2004) have shown that the ideological commitments of the volunteers are divided. We think that FOSS hackers' motives are likely to be mixed and changing over time with the development of software production and business (Weber 2004, von Hippel 2005). There are indications that the majority of the inner core contributors to and maintainers of FOSS projects are employed professionals (Lerner and Tirole 2005). The magnitude and quality of the contributions of the volunteers and the realization of the principle of localized variety are of course dependent on the

motives of the volunteers. Characterizing the nature of the complex of the participants' motives as well as their ideological commitments is an empirical task. We find plausible Lerner and Tirole's (2005) suggestion that with the rapid increase of the commercial appropriation of FOSS, the recognition received in the projects and participants' career expectations will play an important role in the motivation of volunteers.

Moreover, the organizational features of open source projects and the types of licenses adopted have been used as characterizing features of the open development model (Moon and Sproull 2002, Rosen 2005, de Laat 2005, Perens 1999). The research done thus far, however, shows that the organization of the projects changes with time and that different kinds of organizational solutions are seen in different projects, say in Linux and Apache. The organizational form of open development projects is therefore a changing reality and an object of study. Also, different licenses are typically adapted in different projects, and it is increasingly customary to tailor a new version of a license to a specific project. Licensing is a means of regulating the economic interests of the stakeholders and also the participation of the volunteers in the community.

A complementary feature of the open development model might be the modular structure of software or any object of shared distributed construction and its corresponding organization of design. The modular structure enables the distributed contribution and the utilization of localized variety. The contributors can work with limited and specific parts of software without the time-consuming work of familiarizing themselves with the software as a whole. In the same way, a parallel release practice is seen as instrumental to open source and distributed creation, since it allows the simultaneous use of the code by users (who use the stable version of the program) and its further development and modification (due to the simultaneous existence of an unstable, experimental version).

In this paper we understand the open developmental model according to its two constitutive features, an open code and the extensive, varied and distributed participation of volunteers. These two features make the development of the open development model as a novel form of software production possible. We suggest that organizational measures that weaken either or both of the two features would deteriorate or limit the developmental potential of the open development model. We will ask to what extent firms are able to utilize the open development model and what kind of obstacles or limitations they face. The key intermediate phenomenon is the motivation of the volunteers, which influences the possibilities of recruiting volunteers for projects.

## GNOME and OpenOffice.org as Hybrid FOSS Organizations

The hybrid projects we will study, GNOME and OpenOffice.org, have different origins. GNOME evolved as a part of the Free Software movement, which rejected proprietary ownership of software, but has nevertheless gained strong corporate backing. OpenOffice.org's foundation is the open source code of a single piece of software, freed and opened by its owner in order to form a new community around it. *Table 1* shows the tasks of the decision-making bodies in GNOME and OpenOffice.org related to technical decisions and firm/community relations.

Hybrid GNOME was established in 2000 with the birth of the GNOME Foundation, which is a steering committee and legal branch of the GNOME project. Novell, Inc. and Red Hat, Inc together form a majority on the Foundation's eleven-member Board of Directors. Decisions about technical details remain in the hands of individual maintainers of software modules. A release team of eight people makes the overall decisions about the inclusion of modules, informed by an open debate on developers' mailing lists. This decision-making structure provides a consensus-based general design for the desktop as a whole, while giving the maintainers of individual modules the ability to make independent decisions on technical details.

Sun Microsystems made a decision in 2000 to replace the aging standard desktop in their version of Unix with the more advanced GNOME Desktop. Sun joined the GNOME Foundation and started closely collaborating with the project, offering resources and conducting usability studies. In 2003, Novell acquired the leading commercial vendor of GNOME software, Ximian, Inc., to become one of the most influential GNOME companies. The sponsoring companies employ many of the key developers and have a strong representation in the Foundation. The collaboration of employed and volunteer developers thus is a key issue for the overall functioning of the project. We will see later an example of a clash of interests, or a misunderstanding of the rules, between a commercial newcomer in the development effort (Novell) and the GNOME developer base.

**Table 1** Decision making in GNOME and OpenOffice.org in November 2005

	GNOME	OpenOffice.org
<i>Bodies for general project management and mediation between interests</i>	<p><b>Board of Directors</b></p> <ul style="list-style-type: none"> <li>• 11 members (8 employed by GNOME companies, GNOME Foundation's Executive Director, 2 unaffiliated)</li> <li>• legal and organizational support, marketing</li> </ul>	<p><b>Community Council</b></p> <ul style="list-style-type: none"> <li>• 9 members (3 Sun representatives, 4 volunteers, the CollabNet-employed Community Manager)</li> <li>• conflict resolution, project</li> </ul> <p><b>Core Team</b></p> <ul style="list-style-type: none"> <li>• 3 members (2 Sun release managers, CollabNet-employed Community Manager)</li> </ul>
<i>Bodies and persons for technical decision making</i>	<p><b>Release Team</b></p> <ul style="list-style-type: none"> <li>• 8 members (5 employed by GNOME companies)</li> <li>• decisions about inclusion of modules in the final GNOME desktop product</li> </ul> <p><b>Maintainers of GNOME software modules</b></p> <ul style="list-style-type: none"> <li>• 68 maintainers (one third employed by GNOME companies, one third other professionals)</li> <li>• decisions about technical details in modules</li> </ul>	<p><b>Engineering Steering Committee</b></p> <ul style="list-style-type: none"> <li>• 7 senior developer-members (2 Sun developers, 5 volunteers, 1 from Red Hat, 1 from Novell)</li> <li>• advisory committee for the Community Council in case of conflicting interests</li> </ul> <p><b>Project leaders</b></p> <ul style="list-style-type: none"> <li>• 129 leaders and co-leaders from sub-projects (core technical projects: 42, including 12 from Sun; test projects: 16, including 3 from Sun; peripheral Native Language projects: 71, including 15 from Sun)</li> <li>• decisions about technical details in modules</li> </ul>

Wishing to become a more independent desktop solution provider, Sun Microsystems acquired the StarOffice suite from its German developer, Star Division, in 1999. It first distributed StarOffice free of charge. In 2000 it opened

up the source code for the suite in order to benefit from the open development model. This move secured a large Linux user base for Sun. The symbiotic relationship between Sun and the open source developers is based on the fact that a full-fledged office suite is an enormously complex piece of end-user software, which no volunteer community had previously been motivated to produce. Linux users were using proprietary office suites such as StarOffice and Applix Office on their free systems. This did not satisfy the Free Software movement, whose main goal was to produce a completely free computing environment. By acquiring and freeing the StarOffice source, Sun won a good amount of goodwill from Linux users and developers.

To support and guide the newborn community, Sun founded the OpenOffice.org Community Counsel, a similar body to the GNOME Foundation, to guide development efforts, resolve frictions between the community and Sun, and to provide legal support and resources. OpenOffice.org's origin as a piece of proprietary code developed by a team of corporate employees shows in its monolithic design. In an attempt to modularize the code base for easier volunteer participation, the Openoffice.org umbrella project has been split into 88 sub-projects whose leaders decide on technical details. However, the project's asymmetry is evident in the high-level decision making. Long-term guidance is assigned to the Community Council. Among the eight members of the Council is only one volunteer community member. For technical design decisions, the Council consults the seven-member Engineering Steering Committee, the members of which are appointed by the Council from among senior OpenOffice.org developers. However, the Council can overrule the Engineering Committee's advice. Sun dominates the Council and maintains control over the general development of the office suite as a whole, being the originator and a rightful key member of the project. Sun's problematic patronage manifests itself in a mailing list discussion concerning internal Sun development and volunteer OpenOffice.org developers which we will analyze below.

The situation is different in GNOME, which originated in the Free Software community, and where there are several competing companies represented in the Foundation. Despite the differences in origin and current balance of power, the projects share similar problems in their attempt to combine two historically different cultures and modes of organization. We will present examples below, illustrating how contradictory points of views regarding development models and ownership manifest in practice and how the parties in each community work towards resolving them.

We analyze two events in which the tensions between volunteer communities and sponsoring firms become visible. First we examine Novell's attempt to push GNOME's development towards a platform more suitable as a

basis for the company's commercial products and the community backlash that followed. Then we will analyze the demise of a subproject in OpenOffice.org. Examining arguments in a debate on the project's development mailing list, we try to trace the reasons for the failure in building a community around a commercially oriented product.

## **The Desktop Integration Bounty Hunt: Who Decides What Features Are Important?**

The Desktop Integration Bounty Hunt is an initiative by Novell to improve the integration of popular applications into the GNOME desktop. The idea is to offer a prize money for implementing specific GNOME features, such as the integration of the GNOME calendar applet with Novell's Evolution groupware suite or the Mozilla Firefox Web browser with Gaim, the GNOME Instant Messenger. Novell donated the assets for the bounties and paid a nominal administration fee to the GNOME Foundation to handle the money. The individual bounties are usually from \$100 to \$2500 for tasks ranging from implementation of E-mail templates in Evolution to drag'n'drop file transfer in Instant Messaging.

The motivation behind Novell's initiative is interoperability between the core desktop and third-party applications. Novell and other commercial members of the GNOME foundation benefit from the GNOME project's development work by using its code base as a platform to build their own desktop products. They combine the basic open source platform with proprietary software, a combination of other open source applications, and corporate branding. For these companies, interoperability with the GNOME desktop components and third party applications such as Web browsers and office productivity software is of great importance.

For the core members of the developer community, however, the most important issue is the coherence and uniformity of the desktop itself. Therefore, maintainers of GNOME components are by no means always responsive to feature requests or new code contributed by companies or new developers. The proposed code may not directly benefit the basic functionality of the software in question, but maintainers may instead view it as unnecessary complication of their code base.

Bounties were first issued in late 2003, and again in August 2004, and a lively e-mail discussion occurred after both announcements. There were some communication failures on the part of the Board and Novell at the time of the 2004 announcement. The resulting discussion evolved into a lively debate on a few relevant GNOME mailing lists. The main discussion occurred mainly on

the foundation-list, but also on working lists such as desktop-devel-list. Below, we will analyze the 2004 discussion. It clarifies the nature of the tension between the maintainers of GNOME modules and Novell's attempt to push the desktop platform towards a better commercial product for the company.

The 2004 bounties were announced by Novell on the gnome-announce-list on August 5<sup>th</sup>. All the unclaimed bounties from the previous rounds, and a few new ones, were announced. Unlike what had happened previously, this round of bounties had no deadline, but the bounties do not expire until someone collects them. What was confusing was the wording of the announcement's opening: "GNOME Foundation is proud to announce the relaunch of the open source desktop integration bounty hunt."

Immediately after the announcement, a lively discussion arose on the Foundation mailing list. Of the 44 messages concerning Novell's Bounty Hunt, 17 were mainly inquiries about the details and rules of the contest, and answers to those inquiries. Each of the remaining 27 messages contains one or more utterances concerning the initiative. An utterance is the unit of this analysis, and they are analyzed and classified based on their content into four categories, as shown in *Table 2*.

The first type of argument arose from the fact that a Novell representative announced the contest in the name of the Foundation: "The GNOME Foundation is proud to announce the relaunch of the open source desktop integration bounty." This raised a question about the Foundation's role: if the contest is a Novell initiative, why is the Foundation announcing it? Why is the Bounty Web page on the gnome.org servers instead of Novell's own? As the discussion progressed, this confusion was explained as a simple mistake on Novell's part and a communication failure. Eventually, Novell's representatives agreed to make the issue clearer in the future and to move the initiative's Web page away from gnome.org.

The second type of argument took a different point of view of the same problem. If the Foundation were to promote the Bounty Hunt, the task list must be confined to tasks that directly advance GNOME technology. Leading members of the project are concerned primarily with core GNOME technology and applications. Directing resources to aiding application development outside the desktop environment is a commercial effort, not to be associated with the GNOME project. This problem would also be solved by labeling the initiative more clearly as an effort by Novell.

**Table 2** The types of arguments in the Bounty Hunt discussion of August 2004

Type of argument	Number of utterances	Number of speakers	Example of the type of argument
1) The relationship between the Foundation and Novell must be clearly defined	10	8	“Fine, it’s a Novell thing, but could we please make it look like a Novell thing?” (Volunteer)
2) Bounties should be specific to GNOME applications	4	4	“As far as I know Firefox isn’t part of the desktop. (...) This looks more like a Novell todo list.” (Employee of a small software firm)
3) Maintainers must be consulted to ensure that bounties are accepted	6	4	“If GNOME officially supports this (...), please first consult maintainers. (Employee of a small software firm)
4) Monetary incentives should not set development goals for GNOME	5	4	“I resented my priorities getting set like that. (...) Overall I just felt that it was wrong for money to be so closely tied to our development process.” (Maintainer, Red Hat employee)

Potential bounty hunters and previous participants raised an important question about the relationship between the rules of the contest and the technical decision power of the GNOME module maintainers. Other arguments dealt with the same issue, but from the maintainers' point of view. The rules stated that in order to claim a prize, the developer must produce a working solution to a problem and have it accepted into the module in question by its maintainer. This created uncertainty among the contestants and an outside pressure on the maintainers' technical decision making. We will examine the arguments in categories 2 and 3 more closely below, because they uncover the tension between Novell and the maintainers and because of the maintainers' central position in the project's power structure.

GNOME is divided into modules, all of which have their own maintainer. Examples of modules include the calculator application, the text editor, the task bar, the file manager, and so on. The maintainer is responsible for providing a working version of his module for every release of the desktop. He also has the

final call on any technical decisions on his module and decides which feature suggestions are implemented or which fix is applied to a bug or usability problem. The normal procedure for getting a problem fixed, or a feature added, is to file a bug report in Bugzilla, the bug tracking database on the Web. Others will make suggestions and write code fixes to solve the problem, and their merits are debated through the commenting feature of the Bugzilla software. After the debate, the maintainer will pick a solution that fits his greater design plans for the program in question or will dismiss the issue as irrelevant. The Bounty Hunt interfered with this decision-making procedure. This came up in two ways in the debate, in arguments categorized here into classes 3 and 4.

Let us first look at the problem from the point of view of a participant in the contest. A developer and an employee of a small software company who had participated in the contest earlier had had a frustrating experience when he had worked on a task issued in the contest and had implemented a solution:

The maintainer is, however, **not** willing to spend any time on it, which means that patches don't get reviewed, applied and the bounty never got awarded.

The fact that I wasted full 36 hours in a row on this [...] with a friend and that, in the end, we didn't even get a single comment from the maintainers of the affected module is very frustrating.

For Novell, this means that the desired feature was indeed implemented, but was never accepted into the code base of GNOME. Maintainers may have various reasons for not accepting the code generated in hopes of Novell's money. The issue was never filed into the bug database, or the feature may not be desired from the point of view of the general design of overall GNOME behavior or the planned feature list of the module in question. One maintainer showed great resentment of outside interference into the maintainer's own vision, even though he did agree to include the code into his module:

The bounty immediately went very high up my priority list. Why? Not because I felt the feature was any more important than anything else on my priority list, but because people who were expecting to get some money wouldn't get it unless I prioritized it. I resented my priorities getting set like that.

Overall I just felt that it was wrong for the awarding of money to be so closely tied up in our development process. "Who decides what features are important? The one with money! How do we get hackers to do something? Offer them money!"

The reason why the contest rules stated that only implementations accepted by the maintainers are eligible for claiming the bounty money is easy to understand. Novell had a list of fixes and features they wanted to be implemented in GNOME in order to have a better base for their own commercial offering of the desktop. The company could develop the features in-house or accept code submissions from the contest and apply them directly to their version of GNOME. However, in doing so they would lose the advantages of the open development model and end up having to modify the code base more and more, essentially developing their own desktop instead of having the community help them with it.

Novell clearly had not taken the power of the maintainers into account carefully enough, since resistance from the maintainers could effectively deny any of the Novell features from inclusion in GNOME. The initiative failed to strike a balance between their own sponsorship and the maintainers' final say in technical matters. The maintainers are committed to building the best possible software utilizing the widely distributed and diverse developer and user base. Novell wanted to utilize this resource as well. Novell, however, was unable to guide the construction of the GNOME code base through using the monetary incentives typical of commercial enterprises.

The Foundation Board Members and Novell had to take the concerns expressed in the debate seriously. Apologies from Novell regarding the confusion of the role of the Foundation and itself came up quickly. Novell also explained that the bounties announced at this time were not essentially new and that they had been approved previously by the Board. Novell also decided to make changes to the bounty program, following the suggestions in the discussion. The Bounty Hunt Web site would be moved outside gnome.org. Offended maintainers were encouraged to contact the Novell representative, to work out an agreement or to remove the offensive bounty. In the future, distribution of bounty money would also not be handled by the GNOME Foundation but by Novell, so as to not cause any confusion about the originator and sponsor of the contest. The appropriate maintainers would be consulted in the future before issuing any new bounties. These measures seemed to pacify the situation. However, the main questions about the nature of Novell's involvement in the project through introducing a direct monetary reward for individual features remains. Up until November 2005, new bounties had not been issued, so the implementation of the promised improvements to the initiative cannot be confirmed.

## **The failure of the Groupware project: Sun closes the code**

The OpenOffice.org Groupware project (OooGW) is one of OpenOffice.org's 87 sub-projects. The project was introduced by Sun to the community at the OpenOffice.org conference in March 2003. The aim of the project was to develop an open source groupware application to be integrated into the OpenOffice.org suite. Groupware is software that is designed to allow a group of users on a network work simultaneously on a project. A central server is needed for this to provide different clients with communication services (such as e-mail), group document development, scheduling, and tracking. Documents can include text, images, or other types of data.<sup>5</sup>

The level of participation in the late spring of 2004 on the Groupware mailing list was low. The issue that was being discussed on the mailing list was an end-user application, "Glow." It was supposed to cover such functionality as group calendaring, mail, instant messaging, shared folders, web whiteboard, and peer-to-peer file exchange. It was the only technology that had been developed in the project so far. After the Glow-related discussion was a silent period that lasted up to the beginning of July 2004. The observed low level of communication made us think that the Groupware project was not attracting many members or newcomers. Three Sun-employed developers had contributed code to the CVS (Concurrent Version System) code base.<sup>6</sup> Most of the code was contributed by the project leader.

The silence of the mailing list was broken in the beginning of July 2004 by an OpenOffice.org marketing project member inquiring about the status of Glow version 0.4. The Groupware project lead responded to this inquiry. He explained that Sun had re-organized the Glow team and transferred the code from the public CVS to Sun's internal repository because the pace of development had been too slow. He also stated that there was a possibility that Sun would open the source code of the next version of Glow to give Groupware volunteers another chance to work on it. His announcement activated 26 project members to post 130 messages to the project's mailing list between July 4, 2004 and February 3, 2005. From the flow of e-mails, we chose 47. The unit of the current analysis is an utterance. Utterances in the e-mails were analyzed on the basis of their meaning. One e-mail message can contain more than one argument. The arguments were categorized into three themes, which are presented in Table 3.

---

<sup>5</sup> <http://www.maptrax.com.au/standardscompliance/glossaryoftermsatoh/>

<sup>6</sup> The CVS is used to manage the public source code for most Open Source projects. It enables a large number of developers to simultaneously work on a common code base.

**Table 3** Types of arguments related to the closing of the Glow code on the Groupware project's mailing lists (4.7. 2004 – 3.2. 2005)

Categories and Types of Arguments	Number and Position of Speakers	An example of the type of argument
<b>1. Identity and ownership of project</b>		
1a Groupware is Sun's project, not a community project (5 utterances)	5 Volunteers	"OOoGW is clearly not a community project, it's completely controlled by Sun, that is, what happens in OOoGW is <i>_only_</i> decided by Sun." (Volunteer 6, programmer)
1b Groupware is a community-project (3 utterances)	2 Volunteers and 1 Sun-employed developer	"But you guys <i>*are*</i> the community...- it's your call. Whatever you do, I'll try to pitch in with some of my none too copious spare time."(Sun-employed ex-lead)
<b>2. Interpretations of the reasons that led to the closing of the code</b>		
2a There are not enough volunteer programmers in the Groupware project (15 utterances)	8 Volunteers and 1 Sun-employed developer	"The chance for any OOoGW application to become a successor of ...comparable well-known mail and calendaring client has definitely gone. :( And: This is NOT because of SUNs development, but because of the low engagement of community developers!" (Volunteer 8)
2b There is a lot of talk but no code in the Groupware project (7 utterances)	3 Volunteers and 1 Sun-employed developer	"OOGW is nothing more than a lot of visions and ideas and a half application named Glow." (Volunteer 9)
<b>3. Suggestions related to the future of the Groupware project</b>		
3a Groupware members should wait for Sun to decide whether to release the new Glow code as open source (11 utterances)	6 Volunteers, 1 Sun-employed developer and 1 CollabNet-employed Community Manager	"Seems to be the most realistic option (to wait)." (Volunteer 7)
3b Start a new project based on the older version of Glow (6 utterances)	3 Volunteers and 1 Sun-employed developer	"One wonders why nobody has been doing it (...Start/continue hacking on Glow0.2b ...) so far." (Sun-employed programmer)

The frustration experienced by the project members after they were informed that the Glow source code was no longer publicly available is expressed in the following excerpt:

From a personal standpoint, I have to ask myself : "OK, so why bother helping test this thing, if any of the feedback I put in goes into a "behind closed doors" development workspace just because Sun has to meet deadlines ?" (Volunteer 2, tester)

For this volunteer it seemed that the most important reason for contributing had been taken away with the source code. This volunteer was a potential end-user and tester, not a programmer. The project members' opinions concerning the identity and ownership of the project, which are presented in Table 3, supply other possible explanations for the closure of the code:

OOoGW is clearly not a community project, it's completely controlled by Sun, that is, what happens in OOoGW is *\_only\_* decided by Sun. (Volunteer 6, programmer)

OOoGW is not exclusively Glow. The intent was to have an open community driven project. ...I consider anyone participating in any fashion, major or minor a member of the community. Constructive ideas are always welcome, and will be presented here openly. (Volunteer 12, co-lead)

These comments pose the question about who constituted the Groupware "community": Sun's developers, volunteers or both? (See categories 1a and 1b in Table 3.) Sun's goal was to establish a volunteer Groupware community with a variety of heterogeneous expertise. However, most of the programming had been done by the Sun-employed project lead, which indicated that there were not enough volunteer programmers (see 2a and 2b). The volunteers talking on the mailing list said that they wanted to contribute by testing, fixing bugs, writing documentation and manuals, and translating. However, they could not fully use their skills before there was some working code to test and criticize. Thus, the Groupware community consisted of one Sun developer-representative, and volunteer users anxious for a Groupware solution for their respective firms and organizations. The productive core of the community was missing (See 2a and 2b). The importance of a "substantial core" (Weber 2004, 271) or the existence of available, runnable code (Raymond 1999) can be seen as a prerequisite for successful FOSS development.

Not having enough volunteer programmers (See 2b) meant no volunteer-developed open source code. Having no "openly" developed code implied that

Sun was not getting anything in return for its investment. Having the new version of Glow ready in time was a priority for Sun. Thus, the code had to be developed in-house by Sun's developers.

What options did the user-volunteers have? In theme 3, three types of suggestions were found. These were: waiting (3a), forking (3b) and closing (3c). Several suggestions related to the future of the project were presented by volunteers, but without programmers nothing more than waiting could be done. The only thing the volunteer "community" (which was comprised at the time by four discussants) could have done autonomously would have been to fork the Glow project, that is, take an earlier Glow version and start developing it in a new project:

The available Glow 0.3 sources *are* licensed under LGPL (and SISSL), this can't be revoked. So if the community decides to continue work on this base, it can do so! This is the great thing about free software.  
(Volunteer 6, programmer)

Instead, after discussing on the mailing list, volunteers could not help but wait to see if Sun would release the new Glow code as open source. The volunteer co-lead expressed: "Let's see what Sun offers...". Ironically, the option of closing the project(3c), which did not get any support from the volunteers, turned out to be the future of the Groupware.

The closure of the new Glow 0.4 would not have been possible without the OpenOffice.org dual-licensing strategy. At the time Sun used both its own license SISSL (Sun Industry Standard Source License) and the free software license LGPL (GNU Lesser General Public License) in implementing its proprietary StarOffice suite. Currently, Sun uses only LGPL. LGPL allows it to utilize OpenOffice.org code the same way as before. The ownership of the OpenOffice.org code materializes in the Joint Copyright Assignment (JCA). According to the agreement, Sun owns the OpenOffice.org/StarOffice code as a whole, and each volunteer contributor owns the piece of code they have contributed. Thus both parties, Sun and the volunteer community, can take the OpenOffice.org code that is licensed under the LGPL and develop it separately from OpenOffice.org if they choose to. Sun could close the code because it had the needed manpower (employee-programmers) to work on the missing pieces of the upcoming Glow code. As a result, the new Sun-developed Glow could be incorporated into the next version of the proprietary StarOffice suite.

The case of the Groupware project shows that the potential end-user participants and the firm had different interests in developing the product. Groupware seemed important to the end-users, but not to the programmers. The source code, even when it was open and available, was mainly developed by Sun's developer(s). The closure of the source gradually leads to the closure

of the project. Groupware volunteers were left without a product, the OpenOffice.org product was left without a Groupware solution, and Sun failed to make use of the open development model.

## Conclusions

This paper analyzed two hybrid FOSS projects with different origins and histories. Despite these differences, they seem to share the same type of problems related to the relationship between volunteers and firms. OpenOffice.org originated *top-down*. Sun Microsystems owned a piece of proprietary software and later opened its source code to give rise to a new open development community. Our study of the OpenOffice.org Groupware subproject, however, shows the difficulty of creating a community around an idea of software that does not catch the interest of voluntary programmers. The project was interesting to the end-users, who showed a willingness to test and debug the program as well as to use it. However, no programmer community was formed to implement any working code to realize the program. Consequently Sun, the originator of the project, ended up closing the source code for the program and developing it in-house. Sun's attempt to develop the program using open development failed.

GNOME as a hybrid was formed *bottom-up*. When the project was founded in 1997, it was already a large FOSS community before the introduction of commercial members. At the turn of the century, several companies joined the GNOME Foundation as members of the community and became participants in the project. The difficulties in introducing monetary rewards in the Bounty Hunt initiative showed how difficult it was for one company, Novell, to guide the project in the direction it desired. It failed to benefit from the open development of GNOME in order to improve its own desktop product. Both cases suggest that the constitutive elements of the open development model and firm-based development cannot easily be combined into a new hybrid model.

'Hybrid' has been a popular metaphor used to depict the new organizational forms between science and economic activity and more largely between public and private activities (Hippel and von Krogh 2003, Tuunainen 2004). As a biological metaphor it refers to a genetic fusion of two species that is unable to reproduce. We prefer to think the future of FOSS and commercial development of software as a dynamic tension-laden interaction or dialog between the two. In the interaction, both retain and develop their particular norms, rules and organizational principles. In the case that the existence and activity of these organizations become dependent on each other, the union

might be called a symbiotic relationship rather than a hybrid. The canonical example of such a symbiosis is the case of Linux-based operating system vendors such as Red Hat, which are dependent on the Linux community and other communities that develop the open source components of their products (cf. Young 1999). On the other hand, the Linux effort would not be such a success without the support of firms and the direct contributions of their employees.

To understand better the nature of the “hybridity” of projects, the identity and organizational positions of the voluntary developers should be known better. As Dahlander and McKelvey (2005) point out, sponsoring firms frequently devote personnel to the FOSS projects, and these professionals also become members of the developer community. Such developers do not fit into the classic picture of voluntary user-developers motivated by a personal interest or “hacker ethic.” This is confirmed by looking at the organizational commitments of the GNOME module maintainers. Among the maintainers, roughly one third are employed by the three largest corporate sponsors of the project (Novell, Red Hat and Sun) involved in the further development and commercialization of the project's products. Another third of the maintainers are employed professionals of firms and organizations that use GNOME as part of their activities but are not involved in the end-user software business. Examples of such companies are Baum Engineering, the multimedia company Fluendo, and Fugro Seismic Imaging. The last third includes students and researchers, among others.<sup>7</sup> This suggests that instead of the homogeneous community of voluntary hackers with a specific motivation, the key contributors likely have various, even contradictory interests, motives and commitments. The discussion on the bounty hunt initiative also indicates that a delicate play of power between the interests of the parties involved might be taking place within the GNOME community.

The heterogeneity of user-developers was suggested as one of the constitutive features of the open development model (Raymond 1999, V. Hippel 2003). However, with the development of hybrid projects, the very notion of a user is becoming more complicated. In projects such as Linux and Apache, the user-developers were capable of code-writing. The community was primarily a community of specialist code makers, computer professionals, students and highly skilled amateurs. In the projects oriented towards producing software for end-users, the roles of the developers and users no longer coincides. It is likely that division of labor and the variety of contributions will increase.

---

<sup>7</sup> The maintainer lists and author lists in the modules' CVS repositories were examined in the development version of GNOME in November 2005. Maintainers' affiliations were checked from their e-mail addresses, online journals (weblogs), and other public information on the Internet. These online sources did not reveal the affiliations of all the maintainers.

OpenOffice.org has millions of users, but the core developers are largely professionals employed by Sun. The users of OpenOffice.org are predominantly end-users not capable of or interested in programming. The fate of the OpenOffice.org Groupware subproject showed the problem in this separation. The volunteer members of the project were end-users willing to test and criticize prototypes and beta versions of the program, which would bring quality assurance to any code produced.

Are we witnessing the emergence of hybrid communities that combine the previously separate open development model and firm-based software development? Since a hectic era of experimenting is going on, it is hard to give any definite answers. The results of our study on GNOME and OpenOffice.org thus far suggest that the two methods of software development cannot be easily combined. Rather, they support Weber's (2004) suggestion that we are witnessing a tension-laden interaction between two distinct modes of software production. In this interpretation, the "hybrid projects" do not constitute new communities, but rather the 'interface' or terrain where the two forms of software development meet and interact while maintaining their proper norms, rationalities and ways of organizing the work. It is also the terrain where competing firms seek benefits from co-operation and pursue their own interests. This interface is a stage where the future of the open development model will be resolved.



## References

- Boyle, J. 2003. The second enclosure movement and the construction of the public domain. *Law and Contemporary Problems* 66, 33-74.
- Brooks, F. P. (1982). *The Mythical Man-Month. Essays on Software Engineering*. Reading: Addison-Wesley.
- Castells, M. 2000. Materials for an explanatory theory of the network society. *British Journal of Sociology* 51(1), 5-24.
- Chesbrough, H. W. (2003). The Era of Open Innovation. *MIT Sloan Management Review*, 44(3), 35-41.
- Dahlander, L. and McKelvey, M. 2005. Who is not developing open source software? Non-users, users, and development. *Economics of Innovation and New Technology* 14 (7), 617-635.
- Dalle, J-M. and David, P. A. 2003. The allocation of software development resources in 'open source' production mode. In Feller, J., Fitzgerald, B., Hissam, S. and Lakhani, K. (ed.) *Making sense of bazaar: Perspectives in open and free software*. Cambridge, Mass.: The MIT Press, 297-328.
- Dalle, J-M., David, P. A., Ghosh, R.A. and Steinmueller, W. E. 2004. *Advancing economic research on the free and open source software mode of production*. SIEPR Discussion Paper No. 04-03. Stanford University, Stanford.
- de Laat, P.B., 2005. Copyright or copyleft? An analysis of property regimes for software development. *Research Policy* 34(10).
- Feller, J. and Fitzgerald, B. 2002. *Understanding open source software development*. Harlow: Pearson Education Limited.
- Franke, N. and von Hippel, E. 2003. Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software. *Research Policy* 32, 1199-1215.
- Freeman, C. and Louçã, S. 2002. *As time goes by. From industrial revolution to information revolution*. Oxford: Oxford University Press.
- Hamerly, J., Paquin, T, and Walton, S. (1999). Freeing the Source: The Story of Mozilla. In DiBona, C., Ockman, S., and Stone, M. (ed.), *Open Sources. Voices from the Open Source Revolution*. Beijing: O'Reilly.
- Himanen, P. 2001. *The Hacker Ethic and the Spirit of the Information Age*. New York: Random House.
- International Institute of Infonomics, University of Maastricht. *Free/Libre and Open Source Software: Survey and Study, Final Report*. June 2002. <http://www.infonomics.nl/FLOSS/report/>
- Kelty, C. 2005. Free science. In Feller, J., Fitzgerald, B., Hissam, S. and Lakhani,

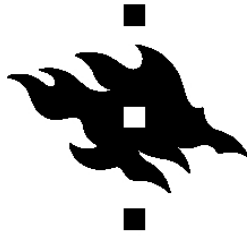
- K. (ed.) *Making sense of bazaar: Perspectives in open and free software*. Cambridge, Mass.: The MIT Press, 416-430.
- Lee, G.K. and Cole, R.C. 2003. From a firm-based to a community-based model of knowledge creation. The case of Linux kernel development. *Organization Science* 14(6), 633-649.
- Lerner, J. and Tirole, J. 2005. Economic perspectives of open source. In Feller, J., Fitzgerald, B., Hissam, S. A., and Lakhani, K.R. (Eds). *Perspectives on free and open source software*. Cambridge, Mass.; The MIT Press, 47-78.
- Lessig, L. 2005. Do you floss? *London Review of Books*. 18 August 2005, 24-25.
- Levy, S. (1994 [1984]). *Hackers. Heroes of the Computer Revolution*. Harmondsworth: Penguin Books.
- McKelvey, M. (2001). The Economic Dynamics of Software: Three Competing Business Models Exemplified Through Microsoft, Netscape and Linux. *Econ. Innov. New Techn.*, 10, 199-236.
- Moon, J. Y., and Sproull, L. 2002. Essence of Distributed Work: The Case of the Linux Kernel. In Hinds, P. and Kiesler, S. (ed.) *Distributed Work*. Cambridge: MIT Press.
- Mowery, D. C., Nelson, R. and Sampat, B. N. 2004. *Ivory tower and industrial innovation. University-industry technology transfer before and after the Bayh-Dole act*. Stanford: Stanford Business Books.
- Nelson, R. R. 2004. The market economy and the scientific commons. *Research Policy* 33, 455-471.
- Perens, B. 1999. The Open Source Definition. In DiBona, C., Ockman, S. and Stone, M. (ed.), *Open Sources. Voices from the Open Source Revolution*. Beijing: O'Reilly.
- Raymond, E. S. 1999. *The Cathedral and the Bazaar*. Beijing: O'Reilly.
- Rosen, L. (2005). *Open Source Licensing. Software Freedom and Intellectual Property Law*. Upple Saddle River: Prentice Hall.
- Shimizu, H., Iio, J. and Hiyane, K. (2004). The Realities of Free/Libre/Open Source Developers in Japan and Asia. *First Monday* 9(11). [http://www.firstmonday.org/issues/issue9\\_11/shimizu/index.html](http://www.firstmonday.org/issues/issue9_11/shimizu/index.html)
- Stanford Institute for Economics Policy Research. *Free/Libre/Open Source Software Survey for 2003*. <http://www.stanford.edu/group/floss-us/>
- Torvalds, L. and Diamond, D. 2001. *Just for Fun. The Story of an Accidental Revolutionary*. NewYork: HarperCollins.
- Tuunainen, J. 2004. *Hybrid Practices: The Dynamics of University Research and Emergence of a Biotechnology Company*. Helsinki University Printing House.
- von Hippel, E. 2001. Innovation be user communities. Learning from open-source software. *MIT Sloan Management Review*, Summer 2001, 82-86.
- von Hippel, E. 2005. *Democratizing Innovation*. Cambridge: MIT Press.
- von Hippel, E. and von Krogh, G. 2003. Open source software and the "private-

collective" innovation model: Issues for organization science. *Organization Science* 14(2), 209-223.

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

Weber, S. 2005. The political economy of open source software and why it matters? In Lantham, R and Sassen, S. (ed.) *Digital formations*. Princeton: Princeton University Press. 178-211.

Young, R. (1999). Giving It Away. How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry. In DiBona, C., Ockman, S. and Stone, M. (ed.), *Open Sources: Voices of the Open Source Revolution*. Beijing: O'Reilly.



UNIVERSITY OF HELSINKI  
CENTER FOR ACTIVITY THEORY AND  
DEVELOPMENTAL WORK RESEARCH

Center for Activity Theory and Developmental Work  
Research

Department of Education / University of Helsinki

P.O. Box 26 (Teollisuuskatu 23)

00014 University of Helsinki, Finland

Tel. + 358 9 191 44165

Fax + 358 9 191 44579

ISBN 978-952-10-3647-7  
ISBN 978-952-10-3648-4 (PDF)  
ISSN 1239-338X