Company Participation in Open Source Software Communities: Measuring Sustainability

Niklas Vainio[1]; Ville Oksanen[2]; Tere Vadén[3]
[1] *Researcher, Hypermedia Lab, University of Tampere, niklas.vainio@uta.fi*
[2] *Researcher, SoberIT, Helsinki University of Technology, ville.oksanen@hut.fi*
[3] *Assistant Professor, Hypermedia Lab, University of Tampere, tere.vaden@uta.fi*

***Abstract* — A framework for analysing the sustainability of a community in four dimensions – social, cultural, legal and economical – is presented. The framework is further differentiated by taking into account the different types of open source software communities, particularly with regard to their work ethics: voluntary or salary-based. In conclusion, the framework is tentatively applied to two communities, Debian and Eclipse.**

***Keywords* — open source software, communities, sustainability, hackers, work ethic, Debian, Eclipse**

I. INTRODUCTION

Most of the contemporary open source software communities are by nature hybrid, consisting of actors with both commercial and non-commercial interests, motivations and backgrounds. The goals of different groups of collaborators, be they hobbyists, volunteers or paid workers, diverge while there is also considerable convergence with regard to the technical goals of a project (see, e.g. [1]). It is therefore good that the old image of a community full of voluntary hackers has ceded for a more realistic approach, which takes into consideration the strongly increasing corporate interest and participation. The work ethic of the corporate world is entering open source communities.

Company participation and the work ethic it implies in communities presents both dangers and opportunities for long-term sustainability. A company needs to be able to identify the systematic variation in motivation, values, ideology and practices between different communities in order to optimize its approach to each of the community intensive projects. A wrong approach may easily prevent the company from reaching its chosen goals and – in the worst case – also severely harm the community itself. Consequently, a general framework for assessing the sustainability and conditions of success of a community of open collaboration will be useful in generating a strategy for interaction between companies and volunteers.

II. FOUR ASPECTS OF SUSTAINABILITY

Free and open communities such as Debian and Eclipse are by nature hybrid, consisting of actors with divergent goals and motivations and from both commercial or non-commercial cultural backgrounds. Companies have increasing interest in collaborating with these communities, which presents both dangers and opportunities for long-term sustainability of a community. In the following, we analyse four aspects of community sustainability: 1) social, 2) cultural, 3) legal, and 4) economical.

A. Social sustainability

The social sustainability of a community depends on the individual characteristics of its members, on its size and form, and the division of labor and power in the community.

Surveys on free/open source software developers show a variety of backgrounds, motivations and values. Multiple motivations are suggested in the literature, including hedonism (“just for fun”), software politics (free software ideals), altruism, identification with a community, peer-recognition, personal technological needs, reputation, learning and working for hire [2], [3]. Different community ideologies can be identified in different projects like Debian, Linux, Eclipse, OpenBSD, Creative Commons and Wikipedia.

Although in the big picture developers seem to be a heterogeneous group, there is more detail when looking at particular projects. For instance, in a recent survey on Debian and Eclipse developers [1], we identified some clear differences between these two communities. Largest differences were that a majority of Eclipse developers are paid to work on free/open source software while for almost all Debian developers it is a hobby; Eclipse developers are on average older than in Debian, and Debian developers are more aware and care about political issues such as copyright and software patents.

The variety of personal characteristics can be either a benefit or risk for sustainability. Eric Raymond's rule “Given enough eyeballs, all bugs are shallow” [4] assumes a large, diverse tester and developer base. In the ideal case, this group includes a whole range of users from less skilled to those with very specific skills, and a wide variety of different use cases, environments and equipment. Variation in aspects like socio-economical status, level of education and geographical location may increase the effectiveness of the group.

Ye & Kishida [5] describe free/open source community participant roles using an onion model. According to the model, at the the heart of the community is often a single person, a project leader, with several supporting core developers around him or her. In middle layers of the onion are active developers, peripheral developers and bug fixers, and on the outer layers users who participate in the

community only by reporting bugs or following and trying to understand the software.

The outer layer of the onion is the largest group and the group size gets smaller as we approach the center. A well-known rule of thumb in sociological research on voluntary organisations is the so-called 20/80-rule (derived from the so-called Pareto Principle), according to which 20 % of the volunteers do 80 % of the work. It seems that software communities roughly follow this rule, with the number of active members drastically reducing with each step towards the core [6]-[9]. Naturally, in a complex or broad project (such as the GNOME), the developer group is divided into several subgroups roughly corresponding with different subtasks.

Krishnamurthy [10] found that 71 percent of hundred most active projects on Sourceforge had five or less developers and 51 percent of the projects had only one project administrator. Projects like these are probably highly dependent on the project leader. The project leader plays an important role also in larger projects such as the Linux kernel. Although Linus Torvalds writes only a minor portion of the new code, he is important for the project as a charismatic leader. The Linux community respects Torvalds' power partially for historical reasons, but also for his skills, and developers see him as essential. In 2002, the community had a “Linus doesn't scale” problem when Torvalds couldn't anymore handle the flow of modifications to the kernel, and a system of trusted “lieutenants” had to be build. Before that, social sustainability of the kernel community was low because of too centralized power and knowledge. With the lieutenant system, (tacit) knowledge is distributed more evenly which makes it easier and more probable for developers lower in the hierarchy to take certain responsibilities if the project leader has to step down for any reason.

Equally important for social sustainability is the system of decision-making and conflict resolution. As a small project grows in time, the diversity of developer opinions and views increases, creating more potential conflicts over technical decisions. The “benevolent dictator” approach chosen by communities like the Linux kernel has been accepted by most, although conflicts have arisen from time to time around issues like the use of proprietary version control system and licensing issues of binary modules. The Apache Software Foundation calls its system “meritocratic” and has a voting system but aims for consensus, which they consider “a very important indication of a healthy community” [11]. It seems that not one system of governance is more sustainable than others – one model does not fit all – but it is important to have a system and it must be suitable for the size and culture of community and do it's job; not too bureaucratic, not too antidemocratic.

In sum, the following heuristics can be used for evaluating the social sustainability of a community:

Increases sustainability:

- diverse user and developer base
- large number developers
- large number of users and contact with the users
- developers in different roles (project leader, core group, bug fixers, bug submitters) with dynamics that encourage learning
- moderately decentralized communication or power structures
- a system of decision-making and division of labour

Decreases sustainability:

- user and developer base with unified background, skills and use contexts
- small number of developers
- small number of users
- no prospective developer base or closed development process
- very centralized communication or power structures
- bureaucratic system of decision-making and division of labour

B. Cultural sustainability

The distinction between social and cultural sustainability is not clear-cut: cultural meanings and artefacts have their effect only as embedded in social practices. However, for the purposes of analysis we may differentiate between the two by noting that the cultural sphere exists, first, on a higher level of abstraction. Cultural aspects of interaction depend on interpretation, language, and coherent patterns of behaviour. Second, the cultural level is temporally different from the social. While some cultural artefacts and meanings may change rather rapidly, there are cultural layers that change little, if at all, during the lifetime of a generation or an individual.

Cultural sustainability of a community is defined by its traditions and history that create and shape its social and ethical norms and practices. While social sustainability is a matter of interaction between individuals, cultural sustainability is something that is created during a longer time period as the community matures. Communities have created documents fixing their position on certain philosophical or technical issues, such as the Debian Social Contract and the related Debian Free Software Guidelines. Changes in practices, such as decreasing openness and transparency, will have a feedback effect on the cultural sustainability.

One of the most famous and important cultural formations discussed in the context of open collaboration is the so-called hacker culture or hacker ethics [4], [12]-[14]. The hacker ethic is thought to contain a loosely interconnected set of values and beliefs that hackers internalize in the acculturation process of becoming active members and contributors of the community. While different author present the tenets of hacker culture in different ways, it seems that the credo "information wants to be free" and the various temporally changing technological ways of promoting this credo ("the Internet

treats censorship as damage and routes around it", "we make the Internet not suck") crystallizes the basic pillar of hacker culture. "The hands-on principle" and a mistrust of all authority and concentration on what is fun or "scratching one's own itch" are other hallmarks of the cultural *ethos*.

However, since the end of the '90s the paradigmatic form of open collaboration, free and open source software development, has seen a radical change in its cultural environment. Through the launch of the open source movement – the explicit motivation for which was to increase the business friendliness of free software – the motivations and institutional background of developers, and consequently, the developer culture, have shifted. In many influential and important OS communities, a significant portion if not a majority of the developers are paid to do the job. On one hand, this has increased the stability and credibility of the communities, thus increasing sustainability, but on the other hand it has brought the traditional hacker culture in contact with the culture or ethics of the "salaryman" who, in the worst case "just works here." For instance, Pekka Himanen has described the hacker culture as a direct opposite of the protestant work ethic ([13], relying on [15]), where stable working hours and hierarchical structures and a clear division of labour with extrinsic motivations for working dominate.

In large and well established communities the clash of the different cultures does not necessarily materialize, as different social arrangements, such as foundations or councils (Eclipse Foundation, Gnome Foundation), can be set for taking care of the interplay of interests. The tension is more eagerly felt in smaller communities in which developers working under the assumptions of hacker culture may – with or without good reason – feel threatened or exploited by a company taking part in and harvesting fruit from the collaborative development work. Correspondingly, a company taking part in open collaboration may consider the unpredictability and uncontrollability of the hacker contingent of the community at least an unpleasant unknown if not an actual risk in itself.

In cases where the company involvement is intense and clear, such as the MySQL community, the risk for sustainability this tension creates is minimal, as practically all responsibility is carried by the company. Mark Shuttleworth has reported of an interesting cultural clash, which involves importing the protestant ethic on hacker culture [16]. In the attempt to boost the development of a software called SchoolTool, Shuttleworth hired a group of hackers for the project. The idea was that given the economical possibility to work exclusively on the software, the team would rapidly augment and enrich the software for which a clear and urgent need was felt. However, the development was slow, if not stalled, because given free hands and ample resources, the hackers did not concentrate on keeping schedules and delivering updates, but started, quite well in line with the hacker culture, to find the best, most robust and sustainable basic structures and architectures for the software. The two cultures did not initially gel in a beneficial way: the logic of paid work did not function in the context of hacker culture.

Over time, communities have developed practices and codified their key principles in documents like the Social Contract of Debian. These documents are often referred to in debates and they produce and maintain the ideological basis of developers. These documents maintain project ideals in the long term and provide common ground for decision-making, and therefore probably have important meaning for the cultural sustainability of the community. (Cf. [17].)

Schematically, then, the evaluation of cultural sustainability of open collaboration may be done along the following heuristics.

Increasing sustainability:

- large volunteer organisation with hacker culture (for example Debian, Wikipedia, GNU, etc.) - well-funded and planned "protestant work ethic" culture (size does not matter so much, for example MySQL)
- explicit foundational documents providing a common ethical ground for developers

Medium sustainability:

- unclear and/or unstable mix of hacker culture & "protestant work ethic" culture (Shuttleworth's story on SchoolTool, the tension between Ubuntu and Debian developers)

Decreasing sustainability:

- small volunteer organisation with hacker culture (risk of losing principal developers; e.g., a majority of dormant sourceforge projects)
- tension between hacker culture and "protestant work ethic" culture (suspicions of being exploited, risk of forking)
- competing ideologies or no common ethical ground.

C. Legal sustainability

In ideal world, legal sustainability should not be an issue for free and open source communities. Unfortunately, the importance of legal risk management has risen sharply during the last decade. The economic significance of software has drawn also the attention of the legal community and as the result the risk of getting sued is today very real. Also the governments are monitoring Internet with closer scrutiny, which means that the questions like taxation and work legislation has to be addressed by the projects.

The “environmental situation” is especially worsened by the fact that both criminal and civil sanctions for IPR-violations have dramatically increased. This is mainly due to IPR-holder's successful lobbying efforts in World Trade Organization (e.g. TRIPS-agreement), EU (e.g. IPR enforcement directive) and the U.S. (e.g. No Electronic Theft (NET) Act) [18]-[20].

To survive in this environment, free and open source communities have to have developed processes and strategies, which minimize the legal risks. Välimäki and Oksanen [21] have developed a framework for risk mitigation, in which five possible options were identified (Table I.)

TABLE I. COMPARISON OF DIFFERENT DEFENSE OPTIONS FOR OPEN SOURCE DEVELOPERS [21]

	Scope	Effectivity	Speed	Price
Disclaimers	Licensees	Low	Fast	Low
Insurance	Market	Medium	Fast	Medium to High
Patenting	Market	High	Slow	High
Avoidance	Market	Medium	Fast	Varies
Lobbying	Regulatory	Medium to High	Slow	Medium to High

The most simple and widely used method is the legal disclaimers. These can be found from virtually all free and open source licenses. However, they offer protection only against claims from the licensees i.e. they offer no protection against 3rd party claims, which are most common in IPR cases.

Insurance is one of the oldest options for risk management. It has been slow to take on in the software sector. Even today, very few companies offer insurance services for the free and open source environment. This is understandable as there would be very few customers i.e. the insurances are typically so expensive that only the richest projects could afford them. On the other hand, there have been some examples of legal defense funds, which are in effect close to mutual insurances. For example, Open Source Development Labs created a special defend fund for:

> The Linux Legal Defense Fund was created to defray legal expenses of Linux end users who may become involved in litigation with The SCO Group on issues that affect the Linux community and industry. The Fund also covers the legal expenses of Linus Torvalds, Andrew Morton and OSDL in connection with the pending SCO/IBM litigation. [22]

Also patenting can be seen as a way to mitigate risks - defensive use of patents is generally useful against patent claims from competitors. Since software patents are typically despised among free and open source developers - and secondly - are very expensive to get and maintain, this option is used rarely. However, certain open source companies like Redhat and Novell have somewhat explored this option.

Another way to reduce risks is avoidance, which covers actually wide set of actions. A very basic example of this could be consulting an attorney for legally unclear matters before making decisions. A strict control on persons, who contribute code, is another typical example, which benefit comes limiting the chance that 3rd party code would added illegally. One quite used strategy is limiting project's scope or innovativeness to avoid liabilities e.g. project may decide that developing p2p-features to its product is currently too risky. Cynically speaking, staying decentralized and poor belongs also to this category, since it make it hard to make profit from legal actions.

The final option in our framework is lobbying for "less hazardous legal environment". This option has been realized recently in certain high-profile campaigns e.g. in the fight against software patents in European Union. The free and open source activists were instrumental in the fight, which ended – against all odds – to the dismissal of the directive, which would have legalized software patenting in Europe.

This more pro-active attitude has been demonstrated also in the more active enforcement of free and open source licenses. In addition to Free Software Foundation (which has been enforcing the GNU GPL), projects like gpl-violations.org have arisen against commercial misuse. Gpl-violations.org defines its goal as:

- Raise public awareness of the infringing use of free software, and thus putting pressure on the infringers.
- Give users who detect or assume GPL-licensed software is being misused a way to report them to the copyright holders. This is the first step in enabling the copyright holders to push for license compliance.
- Assist copyright holders in any action against GPL infringing organizations.
- Distribute information on how a commercial entity using GPL licensed software in their products can comply with the license.

This enforcement is essential for the health of free and open source movement since it helps those companies, which adheres to the rules, against the "bad apples". Indeed, we believe that this enforcement will be even more paramount in the future as the countries with little or no tradition for license compliance turn into biggest development centers of software. However, too rigid control of licensing may also raise the general costs of using free and open source licenses and thus be counter-productive.

Yet another factor for legal sustainability is the choice of free and open source license. A badly chosen license prevents other projects from benefiting from the code, which lessens their interest to contribute. For example, there has been arguments that SUN made a mistake because it did not choose GPL for Open Solaris, which prevented direct code contributions from Linux and vice versa. (e.g. [23]) Furthermore, the compatibility issues may arise also in GNU-world since GPLv3 won't be compatible with GPL version 2.

In summary, the following attributes add to legal sustainability:

- The project uses risk mitigation strategies
- The project's economic footprint is small
- The project is not dealing with legally hot questions like p2p
- The project enforces its rights against misuse
- The license is compatible with the mainstream licenses.

and the following reduce sustainability:

- The project does not have any policies on risk management
- The project has either financial resources or it is causing economic harm to somebody else
- The project is dealing with a legally risky topic

- The project does not care if its rights are being violated
- The license is not compatible with the mainstream licenses.

D. *Economical sustainability*

The very large majority of free and open source projects do not use any other financial resources than the time of their participants. It is therefore natural that most of the early articles on the economics of open source were focusing on the personal motivating factors of the developers. ([24]-[25]). The main theory was that the developers are investing their time because they could get better reputation among their peers – and also among possible future employees. The empirical studies somewhat verified this theory but also found other reasons like personal learning and supporting the goals of free software movement [26].

The most recent economic literature is dealing with the companies' motivations to invest in free and open source projects. The change is very understandable since the projects, which are either started by a company (e.g. MySQL, Maemo) or heavily supported by a company (Mozilla, Google) have become more common and important. It would be fair to say that no firm conclusions exist yet in this area as the companies are still experimenting the co-operation with the community. Ari Jaaksi, Nokia's leader of 770 Internet Table development, lists following reasons for Nokia to use open source:

- Availability of good quality code
- Availability of well-thought architecture and integrated subsystems
- Licensing rules have been decided by the licensee
- No need to execute complex licensing negotiations
- Saving can be up to 6- 12 months in real projects
- The work and credentials of a developer or a subcontractor are open for analysis
- The quality of the people and the components can be analyzed from the source code
- Their willingness to help is easy to verify – just ask
- The activity and direction of the component or product can be analyzed through the project discussions.
- When everything goes wrong – you can take the code and run with it
- Even branch to meet the deadlines (Jaaksi 2006)

In conclusion, the following elements increase economical sustainability:

- The project helps its developers' careers
- The project survives without financial support e.g. volunteers can operate it
- The project is financed by a company, which has a business model.

Similarly, the elements, which most likely decrease the sustainability, are:

- The project does not bring reputation benefits to its developers
- The project is so large/complicated that it requires professional support
- The project can't get support from companies.

III. APPLYING THE FRAMEWORK

We have described above the sustainability framework created based on our experiences with free/open source software communities. In the following, we use the framework to examine two different open source communities, Debian and Eclipse. Debian (www.debian.org) is one of the largest Linux-distributions. It has strong cultural traditions tied to the hacker culture, including a hacker type work ethic [13] and heightened sense for Free Software values. Eclipse (www.eclipse.org) is an extensible development platform and application framework for building software. In contrast to Debian, it has a strong corporate background having been launched by a group of companies including IBM, Borland, SuSE and Red Hat.

Through an analysis of the characteristics of different open source communities, several "ideal types" of communities can be identified. In the survey described above we could recognize two distinct types of community ideology and work ethic. What we call *the hacker ideology* is the traditional FOSS work ethic of freedom, fun and sharing of information, while the opposing ideology is the traditional, *salary-based work ethic*. These two types of ideologies both correspond to certain kind of structures of power and authority. Therefore by "volunteer community" we mean those communities where the hacker work ethic is dominant, and by "company-based communities" we mean the communities where companies and business objectives have more importance and a large percentage of developers are paid for their contribution. In company-based communities traditional hacker values or freedom and sharing have much less importance and participants may not be interested in issues like copyright, software freedom or software patents (identified as "the ideological factor" above).

More detailed analysis and a typology of communities can be created by combining the voluntary/company axis with some other variables. Based on our observations, we provide a preliminary typology of FOSS communities. Four elements are investigated in tandem with the voluntary/company axis: the size of the community, its age and history (in other words maturity), the centrality or decentrality of communication and decision-making in the community and the strength of chosen license.

1) Size of the community. We assume that a larger community is always more efficient and sustainable but potentially increases problem complexity for company participation. The size of the community must also reach a certain minimum size in order to make the open source effect work.

2) Maturity of the community. By maturity we mean the strength of the social and cultural ties, traditions and practices. A mature community is often old in age, and has developed common guidelines and best practices.

3) Communication and decision-making structures of the

community. Different systems of governance exist in free/open source software communities, including democracy, meritocracy and dictatorship. Here we look at how centralised communication is. This tells something about the governance structure, hierarchy and bottlenecks.

4) License. The type of free/open source software license chosen by the community potentially affects who will participate in the community. We classify licenses based on how strong copyleft effect they have. GNU General Public License, for example, is a strong copyleft license, while Eclipse Public License gives more freedom, and licenses like the BSD license are not copyleft at all.

When we combine these four elements with the volunteer/company axis, differences between communities can be identified as can be seen in table II (with examples).

TABLE II. COMMUNITY TYPOLOGY

size / hybridity	***volunteer***	***mixed***	***company***
small	Wordpress		MySQL, Laika
medium	OpenBSD	Mozilla	OpenSolaris
large	Debian	Linux (kernel), GNOME	Eclipse

maturity / hybridity	***volunteer***	***mixed***	***company***
young	Gnash		Laika
developing	Wordpress	Mozilla	OpenSolaris, Darwin
established	GNU, Debian	Linux (kernel)	MySQL

decision-making / hybridity	***volunteer***	***mixed***	***company***
decentralized	Debian		Eclipse
balanced		Linux (kernel)	
centralized	GNU	Mozilla	MySQL

license / hybridity	***volunteer***	***mixed***	***company***
non-copyleft	OpenBSD	Apache	
weak copyleft		Mozilla	Eclipse, OpenSolaris, Darwin
strong copyleft	GNU	Linux (kernel), GNOME	MySQL

In the classification above, we can see both differences and similarities between communities. Based on this analysis, some “ideal types” can be identified which characterise some of the most prominent differences between communities. Four ideal types could be identified:

a) Centralized, company-driven, small community (e.g. MySQL)

b) Large community, several companies, business work ethics (e.g. Eclipse)

c) Large community, several companies, hacker background (e.g. Linux kernel)

d) Volunteer, decentralized, large (e.g. Debian)

Correspondingly Eclipse and Debian have different bottlenecks with regard to sustainability.

To start with Debian, the size and age of the community point out that from the social and cultural perspectives it is very mature. It is very improbable that the community would vanish overnight. On the cultural side, Debian has one of the most developed and explicit guidelines for conduct, The Debian Free Software Guidelines. For Debian, the biggest challenge is that of leadership and decision-making. The community is very large and sometimes the ultra-democratical or anarchic decision making system is felt too slow or otherwise ineffective. On the other hand, it may be precisely this “slow and ineffective” modus operandi that has guaranteed Debian's longevity in the turbulent distro jungle. The hacker volunteers are motivated by group-enriching motivations, and get satisfaction from working together, This binds the community together even in glitches in decision making are sometimes experienced.

On the legal side, Debian has been consistently relying on the GPL, and the principles of free software. This has been one of its hallmarks and may be expected to continue to be so. This formalism has alienated some developers, which have moved to less orthodox projects. In addition, this limits the software the project is capable to offer. For example, the project may not carry the official version of Firefox-browser in the future if the trademark-issues Mozilla foundation are not settled [28]. The project is also in favor of several large companies with big patent portfolios like Nokia, which offers at least implied protection from patent litigation. The project has also well-defined processes for handling alleged IPR-problems.

From the economic perspective Debian also seems well set, as work is mostly volunteer-based. However, the combination of the social and economic perspectives provides a glimpse of the issue that may prove to be most challenging to the sustainability of Debian. Currently the community is dominated by the hacker ethic, but as companies increasingly start using Debian software in their products, they also increasingly employ developers in the salary-based mode. The developers of Debian expressed very positive attitudes towards company participation in our survey, but this may change as the clash between the work-ethics becomes more visible.

Eclipse, on the other hand, is economically sustained by the presence of several large companies. This seems to provide a firm ground in the reasonably foreseeable future. The project has well defined legal guidelines available, dedicated person for legal matters and their license is optimized for the purpose of the project. On the social side the community is fairly heterogeneous, and the project thrives on technological progress. If a rival technology would provide better possibilities, the Eclipse community could face a difficult time.

IV. Conclusion

In this article, we developed a framework for examining the sustainability of open collaboration communities. Based on our experiences from free/open source software communities, we presented four aspects of sustainability which are 1) social, 2) cultural, 3) legal and 4) economical. Using this framework, we looked at two open source communities, Debian and Eclipse. Strengths and potential bottlenecks were identified. In the case of Debian the maturity and established culture of the community are a great asset, as well as a clear legal policy. However, a clash between hacker ethic and salary-based ethic can be predicted in the near future. In the case of Eclipse, the social and cultural heterogeneity of the community makes it more vulnerable to erosion. The promise of technological superiority and progress are the main things keeping the community together, and if the project's progress change, e.g., because of a rival technology, the community may dissolve over time.

References

[1] T. Mikkonen, N. Vainio & T. Vadén, "Survey on four OSS communities: description, analysis and typology", in *Empirical Insights on Open Source Business*, N. Helander & M. Mäntymäki, eds. Tampere: Tampere University of Technology and University of Tampere, 2006. http://ossi.coss.fi/ossi/fileadmin/user_upload/Publications/Ossi_Report_0606.pdf

[2] M. A. Rossi, "Decoding the "Free/Open Source (F/OSS) Software Puzzle. A survey of theoretical and empirical contributions", 2004. http://opensource.mit.edu/papers/rossi.pdf

[3] N. Vainio & T. Vadén, "Sociology of Free and Open Source Software Business: Motivations and Structures", in *Multidisciplinary Views to Open Source Software Business*, N. Helander & H. Martin-Vahvanen, eds., Tampere: Tampere University of Technology and University of Tampere, 2006.

[4] E. Raymond, *The Bazaar and the Cathedral*. Sebastopol: O'Reilly, 1999.

[5] Y. Ye & K. Kishida "Toward an Understanding of the Motivation of Open Source Software Developers". Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, 2003.

[6] B. Dempsey, D. Weiss, P. Jones & J. Greenberg, "Who is an Open Source Software Developer?" *Communications of the ACM*, vol. 45, no. 2, 2002.

[7] A. Mockus, R. Fielding & J. Herbsleb, "A Case Study of Open Source Software Development: The Apache Server", 2000. http://opensource.mit.edu/papers/mockusapache.pdf

[8] S. Koch, & G. Schneider, "Effort, co-operation and co-ordination in an open source software project: GNOME". *Information Systems Journal*, Vol. 12 Issue 1, 2002.

[9] G. von Krogh, S. Spaeth, K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study". *Research Policy*, 2003, vol. 32, issue 7.

[10] S. Krishnamurthy, "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects". *First Monday*, volume 7, number 6 (June 2002), http://firstmonday.org/issues/issue7_6/krishnamurthy/

[11] Apache Software Foundation, "How the ASF works", 2006. http://www.apache.org/foundation/how-it-works.html

[12] S. Levy, *Hackers. Heroes of the Computer Revolution*. London: Penguin, 1984.

[13] P. Himanen, *The Hacker Ethic*. New York: Random House 2001.

[14] R. M. Stallman, *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Boston: GNU Press, 2002.

[15] M. Weber, *The Protestant Ethic and the Spirit of Capitalism*. London: Routledge, 1930.

[16] M. Shuttleworth, "Funding free software projects", 2006. http://www.markshuttleworth.com/archives/4

[17] E. Coleman, "Three Ethical Moments in Debian", 2005. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=805287

[18] P. Drahos, & J. Braithwaite, *Information feudalism: Who owns the knowledge economy?* London: Earthscan, 2002.

[19] M. Pugatch, *The International Political Economy of Intellectual Property Rights*. Edward Elgar, 2004.

[20] A. Bartow, A. "The Hegemony of the Copyright Treatise", 73 U. CIN. L. REV., 2004.

[21] . Välimäki, & V. Oksanen, "Minimizing IPR Infringement Risks in Open Source Projects", in *Software Development - Proceedings of the International Conference on Software Development*, May 27 - June 1, 2005, University of Iceland. University of Iceland Press.

[22] Linux Legal Defence Fund, "Linux Legal Defence Fund FAQ", http://www.osdl.org/about_osdl/legal/lldf/lldf_faq.html/document_view

[23] B. Carver, B. "OSI Shake-Up and Sun's Big Mistakes", 2005. http://sharealike.org/index.php?title=osi_shake_up_and_sun_s_big_mistakes&more=1&c=1&tb=1&pb=1

[24] J. Lerner, & J. Tirole, "The Simple Economics of Open Source". *Journal of Industrial Economics* 52, 2002.

[25] M. Mustonen, "Copyleft – the economics of Linux and other open source software". *Information Economics and Policy* 15(1), 99-121, 2003.

[26] R. Ghosh, R. Glott, B. Krieger & G. Robles, "Survey of Developers. Free/Libre and Open Source Software: Survey and Study, FLOSS, Final Report", International Institute of Infonomics, Berlecom Research GmbH, 2002.

[27] A. Jaaksi, "Building consumer products with open source communities – the Maemo and 770 experiences". Presentation at Linuxworld, Boston, 2006.

[28] M. Gervase, "Firefox Trademark and Debian", 2006 http://weblogs.mozillazine.org/gerv/archives/008347.html