Trends in Free, Libre, Open Source Software Communities: From Volunteers to Companies

Aktuelle Trends in Free-, Libre-, und Open-Source-Software-Gemeinschaften: Von Freiwilligen zu Unternehmen

Jesus M. Gonzalez-Barahona*, Gregorio Robles, Universidad Rey Juan Carlos, Spain

* Correspondence author: jgb@gsyc.urjc.es

Summary The first free/libre/open source software (FLOSS) development communities were composed almost exclusively of volunteers. They were individuals who, despite their affiliation, contributed to the project on their own. They decided which project to join, and their contributions were personal in nature, even when in some cases they were employees of companies with some interests in the project. GNU, the first of such communities, and some others that emerged during the late 1980s and 1990s, followed this pattern. During the 1990s corporate interests started to have a role in some FLOSS communities. Companies hired people from those communities to gain influence, or tasked their employees to contribute to them, again with the idea of influencing their decisions. During the 2000s, corporate communities, in which companies are first-class citizens, have emerged, changing the rules and redefining the role of volunteers and non-affiliated individuals. However, the role of developers, with independence of the company for which they work, is still important even in these communities. This paper addresses this transition from volunteer-based to company-based development communities, and explores the structure and behavior of the latter. ►►► **Zusammenfassung** Die ersten „Free-, Libre-, und Open-Source-Software"-Gemeinschaften (FLOSS) bestanden fast durchgängig aus Freiwilligen. Diese Personen trugen zu den Projekten unabhängig von ihrem Arbeitgeber bei. Auch wenn gelegentlich Unternehmen Interesse an den Open-Source-Aktivitäten ihrer Mitarbeiter hatten, war doch deren aktive Beteiligung persönlich motiviert. Das GNU-Projekt, die erste solche Open-Source-Gemeinschaft, wie auch die nächste Generation an Open-Source-Projekten der 1980er und 1990er Jahre, folgte diesem Muster. Während der folgenden 1990er Jahre entwickelten Unternehmen ein Interesse an Open-Source-Gemeinschaften. Unternehmen stellten Entwickler aus den Projekten ein, um Einfluss zu gewinnen, oder sie beauftragten ihre Angestellten aktiv mitzuarbeiten, ebenfalls mit dem Ziel, Einfluss zu gewinnen. In den 2000er Jahren dann entstanden Open-Source-Gemeinschaften, in denen die Unternehmen selbst als aktive Mitspieler agierten. Diese Entwicklung definierte die Projektregeln und die Rollen von Frewilligen neu. Trotz dieser Entwicklung spielen unabhängige Entwickler weiterhin eine wichtige Rolle. Dieser Artikel untersucht den Übergang von der freiwilligen-getriebenen Gemeinschaft hin zur unternehmens-getriebenen Gemeinschaft und exploriert Struktur und Verhalten dieser neuen Form von Open-Source-Gemeinschaft.

Keywords ACM CSS → Software and its engineering → Software creation and management → Collaboration in software development → Open source mode; volunteer open source; commercial open source; open source community; open source history
►►► **Schlagwörter** Freiwilligen-Open-Source, kommerzieller Open Source, Open-Source-Gemeinschaften, Open-Source-Geschichte

1 Introduction

Richard Stallman started the GNU project, the first FLOSS project in the modern age of FLOSS, in 1983, after quitting his job at MIT. In the first announce [14], he called for contributions from both individuals and companies. But each of them were called for different kinds of contributions: individuals were asked for code, while companies were asked for equipment. He also asked for donations, which would be used to hire developers. Therefore, the GNU project was already considering two kinds of developers: volunteers working in their own time, and developers hired to work on the project. However, he did not call explicitly for companies contributing with hired developers working for them.

One year later the first release of the X Window System [13] was published by MIT. After being licensed to some companies, in 1985 it was released as FLOSS. At that time, several of these companies (notoriously DEC, but also HP, Sun and others) were already collaborating in its development. In January 1988 the MIT X Consortium formed, as a non-profit vendor group serving as an umbrella for the development project. This was the first occasion in which a group of competing companies teamed up to form a corporate FLOSS community (or, as they named themselves, a consortium). Although centered around MIT, which was funded to lead the development effort, participating companies contributed not only with money, but also with their developers.

GNU and X are two early examples of FLOSS projects which show two very different models of development communities. GNU is a good example of a community project driven by individuals, under the umbrella of a Foundation (the Free Software Foundation) established to promote and support the project. X is also a community project, but most of the developers work on behalf of their companies. The MIT X Consortium, their umbrella organization, was the result of an agreement between companies, which were the real actors in the community. Individual developers retained a good degree of freedom with respect to technical decisions, but companies were also present, with their employees having their interests into account.

Still today we can classify many FLOSS projects on one of these two models: communities of individuals and communities of companies. However, most projects are somewhere in between these two extremes. In addition, communities of individuals may have different relationships with related companies, while communities of companies may also have different policies with respect to individual developers.

To better understand all these cases, we propose a complete characterization of FLOSS communities according to how companies and individuals interact in them, based on the analysis of several dimensions. We will also explain how these characteristics can be linked to other aspects of the projects, such as their evolution or their resiliency to changes.

2 Participation of Companies in FLOSS Communities

Although it happened early, the case of X was rare in FLOSS communities of the 1980s and early 1990s. Most of the FLOSS projects of the time were run by, and composed of, individual developers. Companies still had a role, since many of those developers were actually hired personnel, and contributed to the project during their work hours [7]. But since companies were not really a part of the community, this role was subtle, and in some sense more difficult to understand. However, tensions aroused, in some cases even leading to community splits.

An early case of a FLOSS project with some corporate involvement is illustrative of this complex relationship. The GCC compiler was one of the flagship developments of the GNU project. It had in fact its own development community, as many subprojects in GNU do. Some of the developers in this community were working for Cygnus, a company established in 1989 to provide commercial support for GNU software, with some specialization in GCC. Since then, the strategy of Cygnus started to be a component of the GCC strategy, via their developers in the project. This helped to maintain and improve GCC, since in many aspects, what was good for GCC was good for Cygnus.

But eventually, the interests of different parts of the community diverged, and Cygnus led the EGCS fork. They wanted a more functional system, while others, including the Free Software Foundation (FSF), favored stability over new functionality. EGCS was developed independently for several years, until 1999, when the current version of EGCS become again GCC. In EGCS most developers were affiliated with companies which payed them to work on the compiler. After being merged back in the GNU project, this has remained the same. In March 2013, 11 out of 13 members of the GCC steering committee are affiliated with companies [6].

Therefore, GCC was born as a project run by independent developers, with a strong role by an umbrella foundation, the FSF. But as it grew in industrial importance, and companies started to offer commercial support for it, and relayed on it for their business, increasingly more contributors were affiliated with those companies. This caused tensions, to the point of breaking the community in two. The most corporate part of it moved to a new project, which after some time established their own mechanisms to ensure neutrality, and prevent that any dominant company got control, the most visible of which was the self-appointed steering committee. Persons belong to this committee as individuals, and not as representatives of their companies.

Many other projects followed a similar path from volunteer to mostly-corporate communities as the software they produce becomes more used by and more important for companies. However, this is not the only trend: some projects with a clear corporate origin, such as Mozilla or

OpenOffice.org have evolved to be more inclusive with individuals, as will be shown later. Today, probably all communities of a certain size have developers affiliated with companies, but those companies have very different statuses and influences in the corresponding projects.

This relationship between companies in FLOSS communities, and between companies and FLOSS communities, has been studied in the past. One of the first observational studies in the field [3] already realized how complex these relationships can be, classifying the approach of companies towards FLOSS communities in three groups: symbiotic, commensalistic, and parasitic. However, that study does not really enter into the relationships within the project, but focuses on how companies try to obtain benefit from it. From that classification, our interest in this paper is mostly related to the symbiotic companies, since they are the ones interested in supporting the project, and therefore could put some developers to work on it.

Fitzgerald identified as a new paradigm the case of companies collaborating with practices similar to those of "classical" FLOSS projects [4]. His analysis shows how companies can be the primary actors in FLOSS communities, and how that changes their rules, since corporate interests have to be taken into account explicitly. He also describes a type of FLOSS community led by companies, which use FLOSS as a new way of establishing alliances among them, sharing some aspects of their strategy and business lines. These companies enjoy a certain control on the evolution of the project and the resulting software, which is in itself a reward for them.

The relationship between participation in FLOSS communities and performance improvements for companies is analyzed quantitatively in [15]. It shows how, when certain conditions are met (large companies, or companies intensive in R&D), performance improvements can be large when companies engage extensively in the development effort. These improvements explain in part, from a self-interest point of view, why companies are interested in what could otherwise be seen as an altruistic action. The results of their development efforts may benefit any other company that uses the resulting software without devoting any resources to the project, but they will not experiment those performance improvements. Some other studies [1; 8; 17] describe some other strategies that companies use to benefit from this relationship, which further explains their interest in joining or helping to form FLOSS communities.

Another trend explaining the increasing involvement of companies in FLOSS communities is the commodification of industrial software in the form of FLOSS [16]. In many cases, FLOSS components are substituting COTS (components off the self) software, and companies with a strategic dependence on those components have interest in helping to ensure their improvement and maintenance, and to influence their evolution, by participating in the corresponding communities.

A more comprehensive description of the reasons for companies to join FLOSS communities can be found in [11], which can be summarized in cost savings, increased profits per sale, a higher number of sales, and a larger addressable market.

But the relationship does not only benefit companies. FLOSS projects with corporate participation tend to be more popular, and have a better structural quality [2], in addition to the obvious availability of resources and developers contributed by companies.

3 Dimensions to Characterize FLOSS Communities

As seen, companies participate in FLOSS communities because of their own interest, and they are accepted in these communities in various roles because they also benefit from this collaboration. However, having companies in FLOSS projects also causes some problems:

- Tensions between a company and the community. The objectives of a company may diverge from those of the community as a whole, or of a significant part of it. In this case, both parties are interested in finding a common ground in which the community benefits from the resources and development effort contributed by the company, while the company still can push towards its objectives. How, and to which extent, companies are taken into account in a FLOSS project reflects how it has found that common ground, and has a great impact on its community architecture.
- Tensions between companies. The participation of several companies in the community, some of which may be fierce competitors, may cause tensions between them for the control, or perceived control, of the project. Usually, rules, policies and bodies are established so that the community remains neutral in these situations. The different ways of ensuring this neutrality, and to which extent neutrality is sought as a target, lead to different structures for the communities themselves.

When those tensions cannot be handled by the community, the project may be forked, as the history of GCC showed. Therefore, all successful communities have an architecture capable of handling them. However, these architectures are different, since tensions may be addressed in different ways depending on the number of companies involved, their characteristics, the importance of other actors, the history of the community, its size, and some other factors [12; 18].

We propose four dimensions to characterize FLOSS communities, which help to understand the role of companies in them, and how tensions due to their presence are addressed.

- **Lead**: centralized vs. distributed lead. In some cases, a single actor my be the recognized leader of the community. This actor may be a person, as Linus Torvalds in the Linux kernel project, a foundation, as the Open Document Foundation in the LibreOffice project, or

a company, as was Sun Microsystems in the OpenOffice.org community. When these actors exist, they take the ultimate decisions for the project. Depending on their leadership practices, they may be involved in any important decision, or they just decide when the project is unable to reach a consensus.
In the opposite side of the spectrum, actors are equal with respect to the decision taking process. Decisions are taken by consensus or by some voting mechanism among the relevant actors. This is the case of Debian, where developers recognized as such are the actors, or of Apache, with several levels of elective decision making.
Somewhere between these two extremes lie communities which use some kind of weighted voting, or communities in which some actors have special rights for some decisions. This could be the case of the Apache community, where members of some committees, such as Project Management Committees, have some specific responsibilities and privileges.
A specific case in this dimension is that of a company initiating a project, and investing heavily in it, but wanting to have a collaborating community around it. Usually, this company has to keep a delicate balance between holding some control of the community, to ensure its leadership, and the alignment of the product with its strategy, on one side, and letting other contributors participate in the decision-making process, on the other. The Eclipse community is a good example, with IBM initiating the project, but successfully finding ways for others to have enough decision power to participate.

- **Policies**: formal versus informal policies. FLOSS communities usually develop certain policies over time. When they are small and young, those policies are informal, known by the actors because they developed them. As time passes, some communities establish formal policies, which may include formal institutions to act as an umbrella for the project, but which may also be detailed technical guidelines related to the development process. Therefore, in one end of the spectrum we have communities with no formal policies, and in the other formal organizations, with bylaws and policies based on them. Detailed formal policies without an umbrella organization are also possible, although less common.
  Companies usually prefer formal policies and governance documents, which can be checked by their law and engineering departments, and produce some certainty. Therefore, it is usual that communities involving companies are closer to the formal end than to the other one. A recent example is the OpenStack Foundation, established only two years after the project started, with a complete collection of formal procedures and committees. But in many cases very few formalities are enough. The WebKit community is a good example of a project with very intense corporate involvement but a really minimal set of formal policies.
  The reverse is also possible: some communities in which companies are only in the background may have very elaborate formal policies. Debian in an example: they have an umbrella foundation, a constitution, and many rules and formal procedures. But in general, it is more usual that communities formed mainly by relatively small teams of individuals working in their own time are not that interested in these formalities.
- **Actors**: companies vs. individual developers as main actors, and recognized source of authority in the project. Although developers are in the end the members of the community, companies may be recognized as such (formally or informally), and have certain obligations and privileges, such as electing members of some boards, or vetoing certain decisions. In the other end, other communities have special care in all their members being individuals, and ask them to act independently of the company for which they work.
  Examples of the former case are OpenStack or Eclipse, where companies are recognized in the policies (which are formal in those cases), and some of them have certain privileges to elect members of some boards and take certain decisions. GNOME, on the other hand, recognizes only individuals as members of the communities, and has special provisions to prevent employees of any company being a large part of some committees [5] (for example, no organization can hold more than 40% of the board seats). WebKit is an example in which although companies are clearly the drivers of the project, only individual developers are considered in technical discussions.
- **Ownership**: shared versus individual ownership of assets. Who owns the results of the effort of the community, and the resources at its disposal, is also important. And among all the assets, the code is probably the most prominent one. The ownership of the code can be shared, if all of it is owned by the community as a whole, or individual, if each of the contributors own their own parts of the it[1]. The shared ownership of the produced code is implemented usually by assigning it to an umbrella foundation, such as GNOME or Apache do. In the case of individual ownership, owners may be individuals or companies. For example, in the WebKit project companies own the copyright of the code produced by their employees.
  This is important for several reasons. First, only copyright owners can relicense the code. Therefore, those owning the copyright have in their hands the future licensing policies of the project. Second, only copyright owners can enforce the license. Therefore, only they can defend the project against violations of the

[1] We use "shared" and "individual" in this sense, and not in the legal sense, in which a code licensed together by a number of individuals is a shared code, while eg. foundations are "legal individuals", which own all the code.

license. In the case of shared ownership, some form of copyright transfer is needed, either from individuals (in the case of those contributing in their own time) or from companies.

A similar discussion could be done for other assets, such as trademarks (if any) or computing resources.

These dimensions are not specific to companies, but apply to all kinds of FLOSS communities. However, in the common case of corporate involvement, they help to understand how companies are dealt with.

There are other schemes targeted at specific aspects of the community. For example, both the Open-By-Rule Benchmark [10] and the Open Governance Index [9] explore how open is a community governance. However, to our knowledge, there is no other focusing on the relationship of companies with FLOSS communities.

4 Analysis of Some FLOSS Communities

In this section some FLOSS communities will be characterized according to the previous schema. In some cases, they are subcommunities of larger projects (such as GCC in the GNU project), but they have identity by themselves. The analysis has been done based on documentation and discussions publicly available in the Internet, usually in the project websites. When possible, formal documents by the projects have been used.

4.1 GCC (GNU project)

GCC is one of the longest-living FLOSS projects, started by Richard Stallman in 1985. In its current form, its characteristics are:

- Lead: distributed. In its origins, the lead was centralized, with Richard Stallman as the leader. When Cygnus became a large contributor, and started to diverge from Stallman's goals, the tension could not be handled with this model, and the project forked. Currently, after a remerge, the project recognizes the historical importance of Richard Stallman, but takes decisions in general by consensus.
- Policies: mostly informal. Since it is a GNU project, it adheres to GNU policies, and those of their umbrella foundation, the FSF. But these are only a few, and for the rest, the only formal body is the GCC steering committee, which was appointed as the official GNU maintainer by the FSF. Since official maintainers have a lot of freedom in GNU, and no further formal policies have been established, the project is mainly ruled using informal procedures.
- Actors: individual. Although most developers contribute to the project in their work time, companies are not considered actors in the project. For example, it is clearly stated that membership of the steering committee is personal, and members do not represent their companies.
- Ownership: shared. All code belongs to the FSF. Individuals and companies must sign a copyright transfer for all non-trivial contributions.

GCC is developed currently mainly by companies and research institutions, and has become fundamental for many commercial products. The project has achieved neutrality with respect to companies by putting individual developers in the first place, while the shared ownership has ensured that licensing terms are properly protected when needed.

4.2 The Linux Kernel

Linux is one of the most well known FLOSS projects. Since mid-1990s, companies have contributed most of the code in it, either by putting new developers to work, or by hiring kernel developers to work for them. The characteristics of its community are:

- Lead: centralized. Since the beginning of the project Linus Torvalds had the role of "benevolent dictator". Usually, his management style favors consensus, but he takes decisions when needed. Although he has no formal role, his decisions are respected.
- Policies: informal. Except for some coding standards and contributing procedures, Linux has no formal policies. Most processes are known only by practice, and enforced by senior developers.
- Actors: individual. In general, decisions are taken based on technical arguments. Developers do not represent their companies, and companies have no official role in the project.
- Ownership: individual. Contributors maintain copyright over their work, although all of them use the same license.

Probably Linux is the FLOSS project on which most companies depend. In many cases, this dependence is critical, since complete product lines are using it as the base of their software. Therefore, this is also the community with most corporate interests, and one in which tensions with and among companies are common. However, the management style of Linus Torvalds, together with the general respect of the main developers, has been enough to maintain the project together, and keeping companies as heavy contributors.

4.3 OpenStack

Although still young, OpenStack is a quickly growing FLOSS project, and it is becoming one of the largest ones. It is also one of those with more corporate involvement, and causing more corporate strategies to be dependent on it. It can be characterized as:

- Lead: distributed. Although it was clearly led by RackSpace during its early life, it has quickly evolved towards a distributed leadership, with several companies (Red Hat, IBM, HP and others) deeply involved in it, and a decentralized decision structure.
- Policies: formal. After only two years, the OpenStack Foundation has been established. With the foundation and its bylaws, several formal governance documents, procedures and bodies have also been defined.

- Actors: companies and individual. The OpenStack Foundation bylaws recognize not only individual developers, but also companies as members of the community. In fact there are some memberships (platinum and gold), which are usually companies, who have special privileges. From this point of view, although individuals also have their rights and influence, companies seem to have access to more control of the project.
- Ownership: shared. All contributors have to transfer copyright to the OpenStack Foundation.

OpenStack is close to a "community of companies". Although individual developers also participate in the decision-making procedures, companies have special roles available, which let them influence the project. Clearly, the community is oriented towards corporate participation, and this recognition of influence tries to make relationships between companies more transparent.

4.4 WebKit

The origins of WebKit are in two KDE projects, KHTML and KJS, which were forked by Apple for its Safari browser. Later, Apple decided to publish as FLOSS some other components of Safari, and the WebKit project was born in 2005. Later, other companies such as Google, Nokia, RIM, Adobe and Samsung joined the project. Currently, its characteristics are:

- Lead: distributed. Although the project was initially driven by Apple, now many other companies, headed by Google, take part in decisions.
- Policies: informal. There are only a few documents describing formal policies for WebKit, and those that exist are mostly technical. There are no formal committees or voting procedures.
- Actors: individual. Although companies are the driving force behind the projects, they have no formal agreements, and only individual developers are recognize by their peers to take decisions.
- Ownership: individual. Contributors hold the copyright to their contributions, usually as companies.

WebKit is an extreme case of a very critical project, on which the strategy of important product lines from several companies rely. However, it works as a community of individuals. Maybe this is the reason why it can work, with such as deep involvement of fiercely competing companies: letting developers discuss without representing companies help to keep decisions technical and neutral. Of course companies are influencing the project, but this influence is usually in specific areas (ports) that don't affect competitors. Common decisions affecting all companies are usually taken by consensus.

5 Conclusions

Modern, successful FLOSS communities usually rely at least in part on contributions made by companies. It is also increasingly usual that companies have strong interests in FLOSS projects. Therefore, the relationship between companies and FLOSS communities has become a very important issue for the IT industry.

However, this relationship is not easy. Involvement of companies causes tensions both between companies and communities, and between companies themselves. These tensions are reduced with different community architectures. In this paper we propose four dimensions to characterize those community architectures. This characterization allows for a more detailed understanding of how companies integrate in communities, and shows how diverse are the mechanisms for this integration.

Looking at FLOSS communities from an historical point of view, many of them adapted to increasing company participation over time. Well known projects such as GNU, Linux, Apache, GNOME or KDE started as volunteer communities formed by individuals who were working in their own time. But as time passed, when the project showed its value, companies became involved either by hiring developers with experience in the project, or by allocating some of their own developers to it. But not all projects reacted to this phenomenon in the same way, leading to the variety of community architectures that can be found today.

During the late 1990s and 2000s, communities appeared which were designed from their start to allow for strong corporate contributions. In some cases, such as OpenOffice.org, WebKit or Eclipse, a company published some product they owned as free software. In some others, such as OpenStack or GENIVI, a group of companies decided to collaborate early in the life of the project, or even gave birth to it. Although these scenarios are new, they are using models already found in previous cases, and the same dimensions can be used to characterize their community architecture.

Many details of how all these communities behave are subtle, but very important for the companies participating in them. This is why it is important to have detailed information about how they are actually working, and how they are enforcing the formal or informal procedures that they have established. Knowing in detail some of their characteristics, such as how neutral the are to competing companies, or how much influence specific companies have, is needed as well. Fortunately, these communities usually work in the open, which makes it much more easy to find the data to get that knowledge.

However, the proposed classification has some drawbacks, which future work should address. Among them, the following can be mentioned:

- The four dimensions are not completely orthogonal. For example, Policies and Ownership are interrelated, since in most cases decisions about ownership comes from policies. Some interleaving can also be observed between Actors and Lead.
- Some of the dimensions could be split. For example, Policies refer both to technical policies and legal policies. The former are usually decided by the

developers themselves, with little intervention by companies, while the latter are more interesting for the legal departments in participating companies. Therefore, it could make sense to have them as separate.
- Some methodology is needed for the characterization. For this paper we have used review of public documents and some own judgment to decide on the characterization of the case studies. A more detailed methodology should be defined for making it more predictable and useful.
- Each of the dimensions identified shows how communities have dealt with different tensions. For example, Actors show how the project decided to put more power in the direct hands of companies, or preferred a more company-neutral project by empowering individual developers. However, it is not clear to which extent these decisions worked, or if they had an impact on the evolution of the projects over time. It would be very interesting to further study these impacts, and the influence on the development process, if any.

The dimensions cannot in general be read as degrees of "company-friendliness", although some of them show areas which are more company friendly. For example, as it was mentioned, companies are usually more comfortable with clear written rules, specially in the areas of decision-making and participation. This would mean that communities more formally structured would be more friendly to companies. However, there are clear counter-examples, with the Linux kernel and WebKit being cases in which there is almost no legally-binding documents, despite of which they had a great success in attracting companies to collaborate. The same may be said, for example, about recognition of companies as actors, which usually would make them more interested in the community, but could also make it very difficult for competing companies to deal with the tensions of making decisions at the corporate level in collaboration with their competitors.

To us, it is not clear which forces lead a certain community to move to a certain place in the four described dimensions. In some cases it could be the very history of the project, while in others it is a deliberate decision of the communities themselves, or of the companies promoting them. Further research is needed to decide where there are areas in the dimensions which are better or worst for dealing with certain tensions, or which are more effective for attracting, and sustaining, corporate collaboration.

In summary, the four dimensions we propose can be considered as a first step in the direction of new methodologies for comparing and benchmarking how FLOSS communities are dealing with company participation.

Acknowledgements

The work leading to this paper has been funded in part by the European Commission under project ALERT (FP7-IST-25809), and by the Spanish Government under project SobreSale (TIN2011-28110). We thank the reviewers for their help in improving it.

References

[1] Andersen-Gott, M., Ghinea, G., and Bygstad, B. Why do commercial companies contribute to open source software? *International Journal of Information Management 32*, 2 (2012), 106–117.

[2] Capra, E., Francalanci, C., Merlo, F., and Rossi-Lamastra, C. Firms involvement in open source projects: A trade-off between software structural quality and popularity. *Journal of Systems and Software 84*, 1 (2011), 144–161.

[3] Dahlander, L. and Magnusson, M. G. Relationships between open source software companies and communities: Observations from nordic firms. *Research Policy 34*, 4 (2005), 481–493.

[4] Fitzgerald, B. The transformation of open source software. *MIS Quarterly 30*, 3 (Sept. 2006), 587–598. Published by Management Information Systems Research Center, University of Minnesota.

[5] Foundation, G. Bylaws of GNOME Foundation as of April 5, 2002, April 2002.

[6] Foundation, T. F. S. Gcc steering committee. http://gcc.gnu.org/steering.html, retrieved on July 8th 2013.

[7] Ghosh, R. A., Glott, R., Krieger, B., and Robles, G. Free/libre and open source software: Survey and study. deliverable d18: Final report. Tech. rep., International Institute of Infonomics, University of Maastricht, June 2002. http://flossproject.org/report/, retrieved on July 8th 2013.

[8] Jullien, N. and Zimmermann, J.-B. Floss in an industrial economics perspective. *Revue d'Economie Industrielle*, 136 (2011), 1–27.

[9] Laffan, L. A new way of measuring Openness, from Android to WebKit: The Open Governance Index. http://www.visionmobile.com/blog/2011/07/the-open-governance-index-measuring-openness-from-android-to-webkit, retrieved on July 8th 2013.

[10] Phipps, S. The Open-By-Rule Benchmark. http://webmink.com/essays/open-by-rule/, retrieved on July 8th 2013.

[11] Riehle, D. The economic case for open source foundations. *Computer 43*, 1 (2010), 86–90.

[12] Riehle, D. and Berschneider, S. A model of open source developer foundations. In *Proceedings of the 8th International Conference on Open Source Systems (OSS 2012)* (2012), Springer Verlag, pp. 15–28.

[13] Scheifler, R. W. and Gettys, J. The X window system. *ACM Transactions on Graphics 5*, 2 (Apr. 1986), 79–109.

[14] Stallman, R. New unix implementation: the initial announcement of the GNU project, Sept. 1983. http://www.gnu.org/gnu/initial-announcement.html, retrieved on July 8th 2013.

[15] Stam, W. When does community participation enhance the performance of open source software companies? *Research Policy 38*, 8 (2009), 1288–1299.

[16] van der Linden, F., Lundell, B., and Marttiin, P. Commodification of industrial software: A case for open source. *Software, IEEE 26*, 4 (2009), 77–83.

[17] West, J. and Gallagher, S. Challenges of open innovation: the paradox of firm investment in open-source software. *R&D Management 36*, 3 (2006), 319–331.

[18] West, J. and O'Mahoney, S. The role of participation architectures in growing sponsored open source communities. *Industry and Innovation 15*, 2 (2008).

Received: April 1, 2013

Dr. Jesus M. Gonzalez-Barahona teaches and researches at Universidad Rey Juan Carlos, and collaborates with Bitergia, a software development analytics company. He is interested in understanding free/open source software development, in finding ways to improve its efficiency, and in sharing this knowledge.

Address: Universidad Rey Juan Carlos, Spain, e-mail: jgb@gsyc.urjc.es

Dr. Gregorio Robles is associate professor at the Universidad Rey Juan Carlos. His research interests involve software engineering in free/libre open source software, mining software repositories and technology enhanced learning.

Address: Universidad Rey Juan Carlos, Spain, e-mail: grex@gsyc.urjc.es