Designing for Participation: Three Models for Developer Involvement in Hybrid OSS Projects

Hanna Mäenpää, Terhi Kilamo, Tommi Mikkonen, Tomi Männistö

Designing for Participation: Three models for developer involvement in Hybrid OSS projects

Hanna Mäenpää[1], Terhi Kilamo[2], Tommi Mikkonen[1] and Tomi Männistö[1]

[1] University of Helsinki Helsinki, Finland
hanna.maenpaa@cs.helsinki.fi, tomi.mannisto@cs.helsinki.fi
[2] Tampere University of Technology, Tampere, Finland
terhi.kilamo@tut.fi, tommi.mikkonen@tut.fi

Abstract. This paper reports governance practices of three profit oriented companies that develop OSS software with the help of their respective open development communities. We explore how the companies allow development contributions from external stakeholders, and what knowledge they let out of their internal software development activities. The results lay ground for further research on how to organize openness of the software development process in hybrid setups where the needs of different stakeholders are partly competing - yet complementary.

Keywords: Open source, hybrid open source, community management

1 Introduction

Open Source Software (OSS) projects are based on peer-production, self- organization of individuals, and consensus-based decision making [1]. They offer equal opportunities for both long-term members and newcomers in contributing to the development of the software product [2]. In the OSS environment, interactions and development decisions can take place in the absence of market-based or managerial models [1]. Still, active communities that are independent of commercial influence are rare. When companies engage with OSS communities, their corporate direction must be aligned to that of the community's peer production-based approach [3]. In this hybrid environment, the Bazaar truly meets the Cathedral [4], and the forces that drive the collaboration are partly defined by both.

Hybrid OSS collaborations can form around various kinds of software products [5], emerging organically when companies become interested in existing OSS projects or when the projects are initiated by companies themselves by releasing software source code out with an OSS-compliant license. This can be done with the aim of attracting external contributors, increasing the innovative capability of the company or by attracting a niche market that could later be expanded [6,3]. Engagement with OSS communities can be a means for strengthening brand awareness [3]. The strategy can be especially important for small and medium sized companies, helping to diversify their product offerings, taken that the companies have sufficient resources and technological competencies for

building effective and reciprocal collaborations [7]. As Hybrid OSS communities consist of a mix of companies and independent developers with varying motivations [8], finding a mutually beneficial collaboration model poses various challenges.

Several calls exist for further research on management practises in the hybrid OSS arena (De Noni et al., 2013) [9]. Linåker et al. (2015) pinpointed the governance structures in open environments and the openness of the software product development process as interesting topics for future endeavors [10]. Hussan et al. (2016) invite research on practical implementations of the software development process to complement the current work in the academic landscape [11]. To address these calls, we investigate how three profit-oriented companies develop OSS-based products with their respective open source software development communities. We compare how they allow external contributors to access their development process and what knowledge they let out of the priorities and decisions that shape the future of the software. With this, we hope to provide new understanding on how the hybrid OSS development model can be formed and managed at the practical level and which factors can be used as design elements when organizing the community's collaboration model.

The rest of the paper is structured as follows. Section 2 describes typical characteristics of hybrid OSS communities, shedding light on the different styles of governance that companies can employ towards their communities. Section 3 presents our research approach for studying the collaboration practices of the case companies that are introduced in Section 4. Section 5 presents our findings, whereas sections 6 and 7 discuss and conclude the work.

2 Previous Work

Governance of OSS projects involves the means that are in place to steer the efforts of an autonomously working developer community [12]. The governance structure of a community should support collaboration in between the stakeholders, to enable efficient work coordination, and to create an environment where motivation of participants is fostered so that the community can operate in a sustainable manner [12]. It is not straightforward to create a productive and happy community - its boundaries and affordances must be consciously designed and managed [3]. This can take form in different governance configurations that can be mandated from a governing organization or emerge and evolve slowly as a result of a collective learning process [9].

While in community-driven projects decision-making power is distributed among the communitys members, in hybrid set-ups a strong leader for the development project can emerge in the form of a non- or for-profit organization. This host can act as a sponsor, providing for the infrastructure needed for developing the software [8]. This role allows the host to control the community's activities to an extent [6]. Here, the host entity can define on what premises stakeholders can hold various positions, as well as what privileges and responsibilities these roles bring [6]. Implementing these decisions requires careful consideration of the

versatile motivations of the contributors: if the community's members feel their values are being compromised or that their opinions are not being heard, their motivation can deteriorate [13,14]. Distributing knowledge and decision-making power aptly can help to achieve a symbiotic state where the community sees the company as an active co-developer of the software, rather than a "parasitic" business owner that outsources its selected tasks to the crowd [15].

2.1 Policies and Practices

The governance model of a community can vary from free-form and spontaneous to institutionalized, well defined and hierarchical [16]. While the history of an OSS project is reflected in the ways the community functions, fundamentals of how the community can form and evolve are largely defined by the license of the software source code [6]. A fully open licensing strategy increases the projects ability to attract new developers [6], increasing the attractiveness of the software product in the eyes of its potential users [17]. However, if the open development community is strong, this strategy may risk the host's influence on the software product [6,13], possibly creating an unclear legal relationship between the company and its contributors [18]. This risk can be managed by keeping selected software assets proprietary to the company, creating a gate for the external developers [13]. This allows the company to limit the open communitys possibilities to modify and re-use the software and also to choose in which aspects of the development process it wants the open community to participate in [13].

In autonomous OSS projects, a set of freely accessible online tools are used to coordinate work, discuss its goals and deliver contributions [19,6]. These socio-technical systems constitute boundary objects of the community, helping its members to create personalized views on the status quo of the project. Principles according to which contributors should engage them can be implicitly known and embedded in the functionality the software platforms provide. Also, written guidelines and contributor agreements are used to establish how the community should work and what culture it should be built upon. When the development tools and related communications are openly accessible, the processes of requirements engineering, quality assurance and release management become transparent to the public. This, on its part, allows the external stakeholders to understand and practically influence the decisions that shape the software product. In this context, describing means for practical boundary-setting comprises the main contribution of this paper.

3 Research Approach

Case studies investigate phenomena in their natural context [20] and aim to generate case-grounded theory that can be extended, confirmed, or challenged through replication in future studies [20]. For this aim, we describe a mixed-method case study of practices that three for-profit companies use in building

commercial OSS software in collaboration with their respective, hybrid developer communities. With this, we aim at answering the following research questions:

- RQ1: How can a software process be organized to allow a hybrid OSS collaboration model?
- RQ2: What factors can be used to affect openness of the collaboration model?

West and O'Mahony (2008), emphasize the role of accessibility of tasks and transparency of knowledge as important building blocks of the participation architecture in OSS communities [6]. Inspired by this, a research instrument of nine questions (See Table 1) was composed. First five questions (A-E) address accessibility, whereas the four additional questions (F-I) address what knowledge host companies let out of their internal development process.

Answers to these questions were sought by exploring of freely available online documentation, such as the project's wikipedia and contributor agreements. Validity of the findings was evaluated by interviewing an employee of each case company, using the nine question research instrument as a basis for semi-structured interviews. Each answer was coded with a three-step scale from 0 to 2 where number zero represented that the matter is unconditionally closed from the open community. Number one was set to describe that the topic was open to the community to a limited degree. Number two was set to represent that the matter is fully open to the development community to know about or act upon independently. This triangulation of sources and viewpoints, is hoped to increase validity and fitness of the study design for replication in different environments [20].

4 Case Companies

This study focuses on the collaboration practices of three profit-oriented companies that base their businesses on products created in close collaboration with OSS communities. We describe how the companies have arranged their software development processes to allow receiving contributions from external stakeholders. The first case company, Qt Company Ltd., acts as a single vendor for the Qt software, a framework that can be used to develop applications for desktop computers, embedded systems and mobile devices. The Qt project was established in 1991 and was first incorporated by its original developers in 1993. After several commercial acquisitions, the current set-up emerged in 2014. Qt is distributed as free[3] commercial versions. While the source code of Qt is open for anybody to view, the commercial license grants full rights to develop and disseminate proprietary Qt applications. The company offers support and complementary software assets for its paying customers. Size of the community that develops the Qt framework is approximately 400 persons while the number of application developers ramps up to approximately a million.

[3] GPL,LGPLv3

The second case company, Vaadin Ltd. produces an application development framework that can be used to build interactive web applications that run on most operating systems and browsers. The software (Later: Vaadin) is offered with the Apache 2.0 license. It was initially developed as an add-on for an existing OSS product in 2002 from which an independent release was made in 2006. The development project has since been hosted by a company that offers online training, consultancy and sub-contracting of application projects. Development of Vaadin is dominated by the host company, and an application developer ecosystem of approximately 150 000 individuals exists. This is complemented by a developer community which has produced approximately 600 add-ons for the software.

Our third case is the Linux-based operating system Sailfish OS for mobile devices. Its history originates from a software that was released from Nokia as open source in 2011. The current host of Sailfish OS is Jolla Ltd, a startup that sells both proof of concept mobile devices and distributor licenses for the OS. The technical architecture of Sailfish OS is layered and parts of the software, including user interface libraries, are proprietary to the host company. The rest of the layers entail work from several OSS communities, such as Qt and the independent, community-driven project Mer. The Mer software alone comprises of packages that are using various FOSS licenses such as GPL, LGPL, BSD, and MIT. The company uses several crowdsourcing tactics for acquiring feedback about the quality of Sailfish OS. Even with only three projects, the cases reveal the versatility of possible governance configurations.

5 Findings

The Qt, SailfishOS and Vaadin open source communities were found to share significant similarities. For all, a central company acted as a host for the project, sponsored development platforms and participated in development tasks. All hosts advocated for dissemination of the software and arranged community-building activities. In all cases, the host company controlled contents, scope and timing of the software releases. All shared the practice of coupling an external, peer production driven development community with a company internal development process that was exposed to outsiders to a limited degree. Differences were found in which tasks external contributors can participate in and which parts of the development process was transparent for outsiders. Table 1 and Figure 1 summarize these differences.

5.1 Access to Development Tasks

Host companies welcomed code contributions from the open community to varying degrees. In the Qt project, any person could access the workflow coordination tool[4] and assign oneself for a task. Also the tool that supported the code review process[5] was freely accessible. An approval from two humans was required for

[4] Jira
[5] Gerrit

Table 1. Questions used for evaluating the collaboration models. 0 = The aspect is closed/hidden from the open community. 1 = Open to the community to a limited degree. 2 = The aspect is completely open to the community to know about or act upon.

		Qt	Vaadin	Sailfish
A	Who can contribute to software code?	2	2	1
B	Who can test code contributions?	2	2	1
C	Who can accept code contributions?	2	0	1
D	Who can verify defect reports?	2	1	1
E	Who can impact prioritization of work requests	2	2	1
F	Who knows about the integration process of code contributions?	2	2	1
G	Who knows what is on the product roadmap?	1	1	0
H	Who knows timing and content of releases?	1	1	0
I	Who knows priorities of work issues?	2	0	0

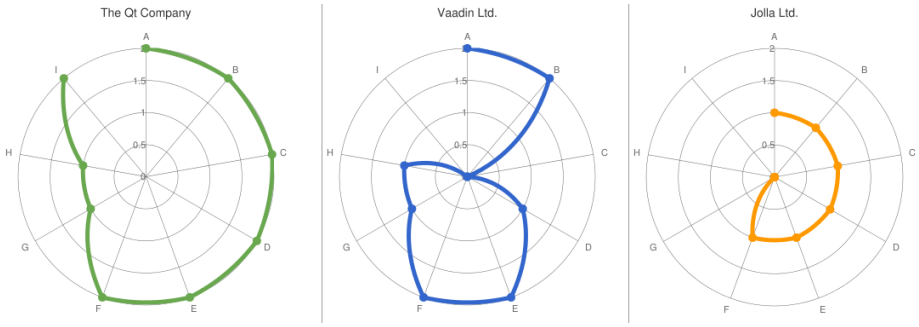

Fig. 1. Comparison of the three models.

accepting a code contribution. Typically the code reviewers were, but were not required to be dedicated maintainers of the software. Partial automation of the testing process facilitated the work of code reviewers.

Vaadin Ltd. presented a more restrictive approach. The work flow coordination tool[6] was accessible for all, yet the company required code authors to write extensive test cases for their submissions. From there, the code was reviewed by dedicated employees of the host company. This arrangement limited community-drivenness of the development process considerably.

Jolla Ltd. did not offer access to work flow coordination tools of the complete Sailfish OS software. Instead it directed contributors to the Mer project's issue management tool[7]. From there, developers themselves needed to identify which

[6] Git issues
[7] Bugzilla

part of the software's architecture their improvement considered, which required the developers to be moderately knowledgeable in terms of the software architecture before being able to contribute. A significant part of the development process was kept private by the host company, which did not let out knowledge about the status of the code integration process. To yield input from non-technical users, the company deployed a question and answer forum where users of the software could report and triage defects related to the user experience of the software, yet without any guarantees on the uptake or their work.

5.2 Influencing Development Priorities

As Qt and Vaadin received most of their requirements from users of their software, their internal development efforts were directed towards fulfilling the needs of their paying customers. In the Qt project, the openly accessible, detailed workflow management system[8] made development priorities clearly distinguishable. Development of Vaadin was largely carried out under the terms of the host company, however input for feature-level priorities were sought from the community by voting on a pre-selected set of features. In the Sailfish project, the development was driven by technical debts that become inherited from the software architectures reliance on the external, independent OSS projects. The situation was largely focused on ensuring that the lower levels of architecture would support decisions made for the proprietary user interface layer.

5.3 Becoming an Actionable Developer

Practices on who can accept code contributions varied in between the projects. In the Qt project, both the core developer- and maintainer teams consisted of stakeholders of many organizational affiliations. These roles could be achieved through gaining individual merit and were open for anyone to pursue. In the case of Vaadin, the role was considered to be open for the community members, yet in practice the strict requirements for contributions made it very hard to enter the role. In the case of Jolla, the MerOS -community was genuinely community driven, welcoming many types of contributions from outsiders. However, the Mer consisted only a fraction of the Sailfish OS, and the host company's software developers occupied the many of the key positions in the community.

6 Discussion and Implications

The various hybrid OSS collaboration models assumed by the companies reflect the mission of the company and its business models. While origins of the project plays a role in how a hybrid community's governance model is built [21], licensing, technical architecture and development tools provide a framework in which the participation architecture of a community can be built [6]. Based on our study,

[8] Jira

we can provide some implications for the factors that can be used to affect the ways in which a host company can manage its boundaries in terms of accepting contributions from open communities.

The participation of external stakeholders can be restricted by licensing [6] or by regulating access to the source code by keeping selected software assets proprietary to the company [13]. Employing a distributed repository strategy proved effective in limiting the possibility of less knowledgeable hobbyists to take up development tasks. Opportunities for becoming a developer can be limited to a specific group of people by requiring a certain level of personal merit and initial or complementary contributions. At an extreme, contributors may be subject to a personal selection by members of the community or even by the host company before becoming an actionable developer.

An important design factor is what tasks a certain role withholds and who are allowed to perform them. Prioritization of tasks can at be the sole responsibility of the host company, or of a selected group of people such as maintainers who have gained their position through personal merit. At an extreme, prioritization can be open to anybody who can access the socio-technical systems or it can be acquired from unrelated, non-technical users by establishing e.g. online customer communities. In all cases, the host company can always define which input from the community it will use in its internal processes.

Community-drivenness requires a critical mass of external contributors [22]. This requirement can be eased by partial automation of the development process such as requirements prioritization or quality assurance. Even though testing and integration processes are supported with automation, it can be required that the final decisions are made in consensus by humans. Limiting this set of people provides a means for adjusting community drivenness of these processes. In this context, strength of the initiation rite for acquiring a certain role could be an interesting topic for future research.

Considering the level of openness: While the Qt company clearly pushed openness to the extreme, its community was found to be the most mature of the three we studied. While Qt started as a community-created project – although arguably with business reasons to do so – the company-created Vaadin project migrated to the open development model only after the it had gained a competitive edge. Sailfish, the newest entrant to the OSS market, was found to be very restricted when dealing with contributions that come from outside of the company. Vaadin represented the middle ground, although they put even more weight on the developers' views regarding direction of development and maturity of the product. In align with Schaarsmidt et al. [21], we find that these company-created projects seemed to maintain more control when compared to their community-driven counterparts.

Limitations. This study dealt with three OSS development communities that were selected to be as different as possible in terms of both age and purpose of their software products. Therefore, the generalizability of the results is limited and further research on projects that are both more similar in nature and of a larger scale is required. The question of representativeness limits the possibilities

to generalize the study results. However, this limitation neither invalidates the findings nor prevents future studies from replicating the approach to evaluate whether they apply fully or partially in other types of hybrid OSS communities.

7 Conclusions

In this paper, we have investigated how hybrid OSS communities can be built to acquire contributors from company -external individuals. These communities, relying on open source software but mixing commercial and volunteer-based activities in their operations, have become an important way to create software systems for various contexts.

Based on the findings, it is clear that companies that use the hybrid OSS development model can act in different ways, leveraging either closed or open practices to support their business interests. A trend in the cases we have studied, it seems as the community matures, the level of openness increases, fostering community-driven peer production. However, the number of companies in the study is far too low to propose conclusive results.

The role of the open development community and the principles according to which individuals can meaningfully participate in the development are essential design factors of the hybrid OSS development model. How these should be defined and managed requires careful consideration on how much knowledge and influence should be released to the open development community, how different roles and privileges can be gained and how autonomously the community should be allowed to steer the development of the software product.

References

1. Y. Benkler, "Coase's penguin, or linux and the nature of the firm," October 2001. [Online]. Available: http://arxiv.org/abs/cs/0109077
2. D. Riehle, J. Ellenberger, T. Menahem, B. Mikhailovski, Y. Natchetoi, B. Naveh, and T. Odenwald, "Open collaboration within corporations using software forges," *Software, IEEE*, vol. 26, no. 2, pp. 52–58, 2009.
3. L. Dahlander and M. Magnusson, "How do firms make use of open source communities?" *Long range planning*, vol. 41, no. 6, pp. 629–649, 2008.
4. E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999.
5. J. Bosch, "From software product lines to software ecosystems," in *Proceedings of the 13th International Software Product Line Conference*. Carnegie Mellon University, 2009, pp. 111–119.
6. J. West and S. O'Mahony, "The role of participation architecture in growing sponsored open source communities," *Industry and innovation*, vol. 15, no. 2, pp. 145–168, 2008.
7. M. G. Colombo, E. Piva, and C. Rossi-Lamastra, "Open innovation and within-industry diversification in small and medium enterprises: The case of open source software firms," *Research Policy*, vol. 43, no. 5, pp. 891 – 902, 2014, open Innovation: New Insights and Evidence. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0048733313001601

8. J. M. Gonzalez-Barahona and G. Robles, "Trends in free, libre, open source software communities: From volunteers to companies," *it–Information Technology it–Information Technology*, vol. 55, no. 5, pp. 173–180, 2013.
9. I. De Noni, A. Ganzaroli, and L. Orsi, "The evolution of oss governance: a dimensional comparative analysis," *Scandinavian Journal of Management*, vol. 29, no. 3, pp. 247–263, 2013.
10. J. Linåker, B. Regnell, and H. Munir, "Requirements engineering in open innovation: a research agenda," in *Proceedings of the 2015 International Conference on Software and System Process*. ACM, 2015, pp. 208–212.
11. H. Munir, K. Wnuk, and P. Runeson, "Open innovation in software engineering: a systematic mapping study," *Empirical Software Engineering*, vol. 21, no. 2, pp. 684–723, 2016.
12. M. L. Markus, "The governance of free/open source software projects: monolithic, multidimensional, or configurational?" *Journal of Management & Governance*, vol. 11, no. 2, pp. 151–163, 2007. [Online]. Available: http://dx.doi.org/10.1007/s10997-007-9021-x
13. S. K. Shah, "Motivation, governance, and the viability of hybrid forms in open source software development," *Management Science*, vol. 52, no. 7, pp. 1000–1014, 2006.
14. S. O' Mahony, "The governance of open source initiatives: what does it mean to be community managed?" *Journal of Management & Governance*, vol. 11, no. 2, pp. 139–150, 2007.
15. L. Dahlander and M. W. Wallin, "A man on the inside: Unlocking communities as complementary assets," *Research Policy*, vol. 35, no. 8, pp. 1243 – 1259, 2006, special issue commemorating the 20th Anniversary of David Teece's article, "Profiting from Innovation", in Research Policy. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0048733306001387
16. P. B. de Laat, "Governance of open source software: state of the art," *Journal of Management & Governance*, vol. 11, no. 2, pp. 165–177, 2007. [Online]. Available: http://dx.doi.org/10.1007/s10997-007-9022-9
17. S. L. Toral, M. R. Martínez-Torres, and F. J. Barrero, "Virtual communities as a resource for the development of oss projects: the case of linux ports to embedded processors," *Behaviour & Information Technology*, vol. 28, no. 5, pp. 405–419, 2009.
18. S. M. Wolfson and M. Lease, "Look before you leap: Legal pitfalls of crowdsourcing," *Proceedings of the American Society for Information Science and Technology*, vol. 48, no. 1, pp. 1–10, 2011.
19. A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 11, no. 3, pp. 309–346, 2002.
20. R. K. Yin, *Case study research: Design and methods*. Sage publications, 2014.
21. M. Schaarschmidt, G. Walsh, and H. F. O. von Kortzfleisch, "How do firms influence open source software communities? A framework and empirical analysis of different governance modes," *Information and Organization*, vol. 25, no. 2, pp. 99–114, 2015.
22. L. Dahlander, "Penguin in a new suit: a tale of how de novo entrants emerged to harness free and open source software communities," *Industrial and Corporate Change*, vol. 16, no. 5, pp. 913–943, 2007.