Opening Industrial Software: Planting an Onion

Petri Sirkkala, Timo Aaltonen, and Imed Hammouda

Tampere University of Technology
{petri.sirkkala,timo.aaltonen,imed.hammouda}@tut.fi

Abstract. This paper studies the problem of building open source communities for industrial software that was originally developed as closed source. We present a conceptual framework for planning the early stages of the release process highlighting the main stakeholders and concerns involved. The framework is illustrated by means of three industrial software platforms reporting first experiences of the community building process. In order to measure the effectiveness of the approach, the use of a quantitative and qualitative evaluation framework is advocated.

1 Introduction

Open Source Software (OSS) is gaining momentum in several forms. In addition to the huge increase in the number of open source projects started and the remarkable rise of OSS adoption by companies, new models of participation in the movement are emerging rapidly [7]. For instance, companies are increasingly releasing some of their proprietary software systems as open source on one hand and acquiring open source software on the other hand. For all these forms of involvement, a central question is how to build and maintain a sustainable community of users and developers around the open source projects.

Research findings show that developing and maintaining online communities in general is a complex activity [14]. In the case of open source communities, the situation is worsened as the problem is multi-facet bringing own kinds of challenges. For instance, this can be viewed as a social question: OSS communities typically come with own kinds of social structures [6,12] that should be tolerated by existing organizational patterns in companies. From a legality viewpoint, the selected licensing type and scheme, for example, can affect the way the open source project is perceived by the community [18]. On the technological and technical side, influential factors include the quality of the software and the availability of support infrastructure [8].

Existing research work on open source communities focuses mainly on studying and analyzing various properties of communities (e.g. [13]). There is, however, little work done in the area of building new communities and sustaining exiting ones in the context of the new rising models of open source software.

In this paper, we study the problem of open source community building in the context of opening industrial software that has originally been developed and used as closed source. In particular, we study the requirements of the community building process

C. Boldyreff et al. (Eds.): OSS 2009, IFIP AICT 299, pp. 57–69, 2009.

and the important issues that should be addressed for preparing a release plan. For the purpose a conceptual framework, which is based on the widely recognized onion structure of open source communities [12], is introduced.

In order to evaluate our approach, the conceptual framework has been applied in the context of three industrial software platforms provided by three Finnish companies. The key findings of the case studies are based on interviews and questionnaires conducted with the companies in addition to our role in the projects as external tutors.

We proceed as follows. In Section 2, we present a general framework for open source community building. In Section 3, we study the problem in the context of three case studies. Section 4 presents an evaluation framework to be used for measuring the success of our example communities. In Section 5, we study related approaches and works. Finally, we conclude with final discussion in Section 6.

2 Framework for Releasing Industrial Software

Building a viable open source community for an industrial software is a complex activity that has to be carefully planned. Prior to the actual release of a software as open source, the publishing company should address a number of important issues which represent essential ingredients to the release plan where actual resources and action points are identified. These issues can be grouped into three categories: the organizational *entities* involved in the release process, the *objectives* of the different parties, and the *nature* of the new born community. We will address each of these three concerns in the following subsections.

2.1 Elements of Release Process

Releasing an industrial software as open source involves multiple stakeholders with possibly different objectives and conflicting interests. It is vital that the building process balances the forces of the stakeholders so that a least minimal satisfaction level is reached by all parties. Also issues like societal norms and legal matters should be considered. Any action not considering these forces could compromise the success and viability of the community.

Figure 1 depicts the main *stakeholders* involved during the early phases of the building process. There are three main groups of stakeholders: the *publishing entity* with its allocated resources for the project, *the industrial partners* and theirs developers, and finally existing *open source communities and other individuals*. As discussed earlier, OSS communities tend to take the shape of an onion with the inner layers taking more leading and contributing roles than the outer layers. The position of each of three groups of stakeholders in the onion structure is determinant by factors like familiarity with the project, objectives, and availability of skilled resources. Compared to the other two groups, the publishing entity is most familiar with the software to be released and most willing to invest in the building process. Thus, the role of this entity is essential and can in fact be viewed as multi-fold.

In addition to the *software* to be released, a number of *skilled developers* from the original development team should be allocated to form the core of the new-born community. The role of the developers is to lead the development of the software and to interface with the rest of the community members as they join.

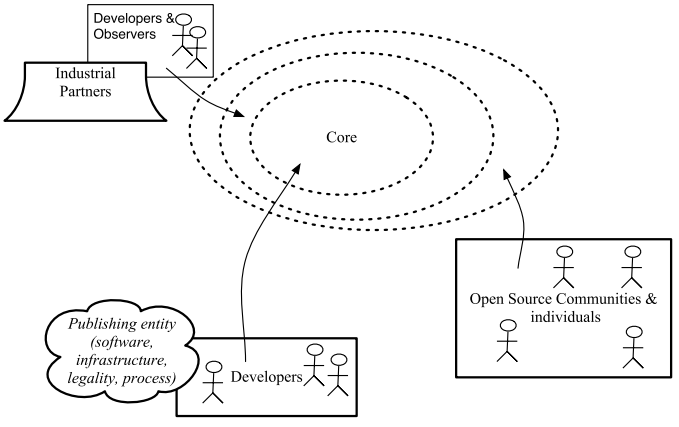

Fig. 1. Elements of OSS Community Building

Furthermore, the entity should provide a proper *infrastructure* for the project to facilitate the planning, coordination, and communication between the community members. The infrastructure should be established according to the principles and work mode of open source communities, thus it may differ from the proprietary model. For the development community to contribute to the project, efficient mechanisms and tools are needed to facilitate the access and management of the project repository. Examples of enabling technologies and tools include website, version control system, defect tracking system, Wiki, mailing list, blog, Frequently Asked Questions, etc.

From a *legality* viewpoint, a license type (e.g. GPL versus LGPL) and a licensing scheme (e.g. single or multi licensing) should be decided as this may affect the way the open source project is perceived by the community. The license decision should take into considerations issues like popularity of the license, compatibility with other licenses, business restrictions. Also, the source code should be legally cleared before being released. This includes copyright and IPR issues. Furthermore, the availability of trademarks and names should be checked. Often, names given for software projects and products are not problematic as long as they are used internally but may conflict with existing same names as soon as they become public.

The publishing entity should also define a *process* through which the elements of the infrastructure are coordinated to guarantee access to the open source project including access to source code, contributing to the project (patches, wish lists, bug reports, bug fixes, documentation, and developer support) and reaching other community members. Finally, the publishing entity should be aware of the dynamics of open source communities. It is important that the existing bureaucracy and organization structures do not conflict with the "‘ release early, release often"’ principle of open source.

In case the publishing entity has industrial partners associated with the software to be released, those partners might also be interested to get involved in the community. Typically, there are two kinds of partners: enthusiastic and conservative. Enthusiastic partners are in favor of releasing the software as they see it as an opportunity for their business. Betting on the success of the community, these partners usually participate

with developers to contribute to the development of the software and stay close with its evolution. Therefore, those developers are closer to the core team and might have key roles in the community.

Conservative partners however are reluctant for the software to go open source as this may change their mode of operations and business. Still, those kinds of partners would like to observe the evolution of the software and the outcome of the community by having own members in the community. Though familiar with the software, those observers are closer to outer layers of the onion structure as they don't assume any key development role in the early phases. It is of course possible that there are no industrial partners involved when releasing a software. In this case, the publishing entity should replace the void in the middle layer(s) of the onion structure by allocating more of their own resources and get other companies interested to the project.

The other vital element in the community building process is the existing open source communities and other individuals. The individuals could as well represent the interests of companies that are not partnering with the publishing entity. The software to be released should be carefully introduced to this group because existing communities and other individuals represent a pool of potential contributors who could join the project if they get interested and motivated. It is possible that individuals joining the new-born community do not have any earlier experience with open source projects. In any case, the new comers typically join the community as passive users and then may take key roles as they show commitment and value, thus penetrating the onion structure inward from outer layers. For this to happen, a well-defined developer promotion policy needs to be in place.

2.2 Objectives for Releasing Software

Objectives in industrial open-source setting has been studied for example in [4]. Traditionally the motivations of the stakeholders can be divided to *intrinsic* and *extrinsic* ones [4]. The former include motives like hedonic, political issues, altruism and peer recognition. Examples of the latter are direct payment, learning, user needs and reputation.

According to [4] the highest-ranking incentive for companies is that "[OSS] allows small enterprices to afford innovation", getting feedback is the second, and quality is the third one. For more information we refer to [4].

Based on our initial work, we have divided the objectives into three categories: *Marketing*, *Business Models* and *Shared cost*. Another classification could be *quantitative* and *qualitative*.

Marketing:

Boosting the field. The company wants to foster the business around their field of operations or to improve the technology in their business area.

Marketing. Releasing a software as open source makes the product familiar to people, which may facilitate the marketing and sales process.

Recruiting. The community built around the released products provides a pool of potential employees with a technical knowhow of the company's field. The company seeks to recruit skilled members of the community.

Reputation. The company wants to send a message about their involvement in Open Source hoping to gain reputation.

Promotion of the system. The objective is close to marketing.

Business Models:
Software As A Service (SAAS). The company wants to build business model around hosting the released product. The company might also provide services based on the product.

Internal usage. The released product is used in house to support company operations. The goal of the release is to improve and evolve the product by the community.

Networking. The company wants to build a network of companies and communities in the field. The company may exploit the network for operational needs.

Knowledge. Company is interested in learning about the OS ecosystem, the release process, and how the OS will affect their business and operating models.

Consulting. The company wants to build business based on giving consultancy about the released product.

Codevelopment. The company wants to develop infrastructure with other parties, so that everyone can benefit.

Software:
Functionality. The company aims at adding new fuctionality or features to the software through community participation.

Quality. The company's goal is to collaborate with the community to improve the overall quality of the product such as stability, usability, and documentation.

In addition, the publishing company should spell out the objectives, expectations, and concerns of industrial partners and open source communities involved in the release process, reflecting on the reasons why those should get interested in the project. The objectives of the other parties should be regarded as essential factors for the success of the project.

2.3 Nature of the New-Born Community

A central question when releasing a software as open source is the kind of community to be built for that software. For instance, the publishing entity needs to characterize the new born community with respect to its type (company-based, volunteer, or mixed), the dynamics of its onion structure, and the way the community should evolve.

In the case of volunteer communities, developers are contributing to the project for free and are driven by a hacker attitude. In contrast, company-based communities are

more tied to company objectives and most of the developers get paid for their contributions. Communities that mix both volunteers and developers working for a salary are referred to as mixed communities.

The publishing entity should also think about the dynamics of the onion model structure. For example, it is important to decide on how close or open the core group would be. In the case of a closed core, developers cannot make it to the core as this will be controlled by the company. In an open core model, developers may be promoted to leading roles based on on their merits, i.e. their contributions, capabilities, and reputation.

Furthermore, the publishing entity should have vision or expectation on the topology of the new born community in the long run and its relation with existing open source communities. For example, the new born community could be part of an existing community or totally independent. Depending on the nature of the software to be released, the new born community may be planned and built as an ecosystem of related smaller communities. In this case, each small community could focus on a specific part of the project. In the next chapter we discuss three cases of community building for industrial software.

3 Case Studies

In order to evaluate our approach we have applied our *conceptual framework* to three industrial case studies. Two of the cases are in early state of opening and one has been opened before our study. We have queried the companies about the details of the release framework. The answers to our queries have been collected in tables below. Finally the NoTA case is presented in more detail.

3.1 Contextual Framework for the Cases

We applied the framework to the three industry cases that were releasing a software as open source. The cases are introduced here in alphabetical order. Then we present the entities, objectives and nature of the community for each case.

ITMill Toolkit is a web user interface framework for developing Ajax web applications. It was originally developed by a company *ITMill*. The software was released into open source late 2007. ITMill Toolkit 5 uses Google Web Toolkit and comes with Apache 2.0 open source license. Our second case is **Network on Terminal Architecture**, or NoTA. It is a modular service-based architecture framework for embedded devices. NoTA was first developed in *Nokia*. The software was opened in summer 2008. NoTA 3.0 is dual licenced under GNU GPL v2 or with a commercial royalty free licence. Finally **Wringer** is an User Interface scripting engine for embedded devices. Wringer was originally developed by *Sesca Embedded Solutions* and it is to be released under GPL license in 2009.

The *entities* depicted in Figure 1 are listed in Table 1 within each case context. The stakeholder *objectives* given by the companies are shown in Table 2. Objectives are listed as prioritized list and refer to section 2.2. The intended *nature* of the community of each case is shown in Table 3. These tables are based on our queries to the releasing companies.

Table 1. Entities of OSS Communities

Software	Industrial partners	Targeted open source communities
NoTA	Semiconductor manufacturers, Other mobile device manufacturers	Embedded and distributed systems, Device architectures, Open standards
ITMill Toolkit	Customers using product, Consulting clients	Web development, Java (Java EE), Google Widget Toolset
Wringer	-	Embedded systems, Software engineers targeting mobile devices, UI applications, Operational displays, ICT Automation, Javascript

Table 2. Objectives for Open Sourcing the Product

Software	Objectives
NoTA	1. Networking 2. Knowledge 3. Boosting the field
ITMill Toolkit	1. Marketing 2. Consulting 3. Networking 4. Reputation 5. Recruiting
Wringer	1. Recruiting 2. Consulting 3. Knowledge 4. Networking 5. Reputation 6. Boosting the field

Table 3. The Nature of the Intended Community

Software	Community type	Core group	Intended future
NoTA	Company based	Closed	Ecosystem of related communities
ITMill Toolkit	Mixed	Closed	Independent community
Wringer	Volunteer	Open	Closely connected with GTK+ bindings community

Our key findings suggest that networking, recruiting and consulting are the most important objectives. On the the other hand SAAS, internal usage and reputation are the least ones. As for the nature the intended community it seems that the diversity is high.

3.2 NoTA Case in Detail

Nokia has released NoTA software from the *cathedral* of the corporation into the *bazaar* of a new open source community [15]. Figure 2 illustrates the elements in this case. The solid elements represent the state before release or the cathedral state, and the dashed elements stand for the released or bazaar state.

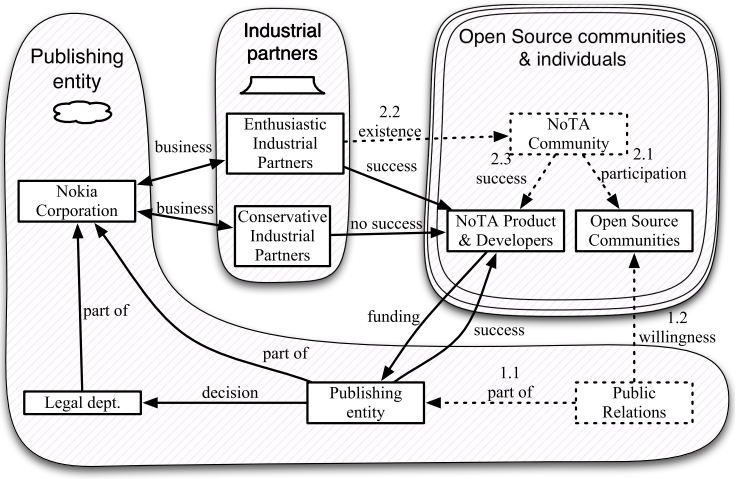

Fig. 2. Elements of the NoTA OSS Community Building Process

In the figure the squares are the instances of entities discussed earlier in section 2.1. The hatched areas are the generalized entity groups. The Publishing entity is *part of* Nokia, in this case a project team. Enthusiastic Industrial Partners, including Semiconductor manufacturers, are doing *business* with Nokia and they also are keen to see the NoTA Product to *succeed*. The Conservative Industrial Partners would prefer to keep the current business as usual and, therefore, they would rather see NoTA *not to succeed*. The targeted open source communities were enumerated in Table 1. The NoTA Community in Figure 2 refers to the onion in Figure 1.

The steps to initialize the community were described in the release plan of NoTA. Let's focus now in the dashed portion of Figure 2. The first step, *(1.1)* in the process of opening NoTA, was to fill the Public Relations position. The Public Relations person needed to be in close co-operation with the developers to ensure the insight of the product was properly channeled to Open Source Communities. The Public Relations depends on the *willingness* of the Open Source Communities *(1.2)*. The Public Relations needs to motivate the Open Source Communities by making positive perception of NoTA and Nokia. This requires transparency to make clear that the they have no hidden objectives [9, p. 164]. Any monetizing attempts need to be carefully thought to avoid bashing the community [9, p. 164]. The methods include sending positive tone messages. Also it requires being prompt in answering any queries.

The next step *(2.1)* is to have the members of open source communities to participate in Community. This yields to networking of developers which in turn strengthens the Platform Community and achieves Nokia's objective of networking. The developers that belong into multiple communities act as bridges between projects [19] and might influence other communities into joining efforts. Participation can be endorsed by building community infrastructure for communication, bug reporting, downloads etc. Defining the processes to incubate other projects and policies to penetrate the onion model will also help members to participate [19]. Core developers must communicate with the community through the infrastructure [9, p.164]. Participation can also be promoted

by contests and academic projects. Existence in *(2.2)* can be strengthened by having organized structure in community and stating clearly the goals and objectives of the community. Success of NoTA *(2.3)* can be endorsed by Nokia showing keen interest in it and organizing conferences, competitions and academic publications around the product.

4 Evaluation

Subsection 2.2 enumerated possible objectives the company may have for the opening the software product. In this section the conseptual framework is augmented with evaluation of the success of the opening. First a number of low-level measures is enumerated, and the we sketch a model for refining the measures for assessing whether the company is reaching it's objectives or is it moving to wrong direction.

4.1 Measures

The measures presented here are abstract, and they must be instantiated to concrete projects manually. They can be divided into two categories: community and software. The former are related to the growth and sustainability of the developing community and the latter measures deal with actual product.

Community-related measures:
The number of contributors measures the number of people close to the core of the onion.

The number of users subscribed to the mailing list measures the number of developers who closer to the core of the onion, or are willing to penetrate towards the core.

The amount of requests, feedback or inquiries received tells about the number of interested people at the different layers of the onion model.

The geographical distribution of the community members is high if the marketing of the project has been successful, and the project can be considered as a sustainable one.

The web-site access is related to the number of people in the onion.

The number of media hits measures the success of marketing.

The number of people participating in project events and meetings is about the activity and sustainability of the project. Developers who are active enough to travel to events are loyal to the project.

The number of views for the social media, like FaceBook or Youtube, tells about the success of viral marketing.

The number of scientific publications describes the academic interest to the project.

Software-related measures:

The number of downloads of the software tells about the overall interest to the product and community. The measure includes both first-time downloads and updates of an older version.

The number of reported bugs and feature requests indicates how interesting is the product.

The number of projects built on top of the platform is big when the project is part of a larger network of projects.

The number and the impact of contributions received measures the software engineering activity in the developing community. For example, the impact is large when the contribution (i.e. patch) touches a large number of modules, or big portion of the codebase.

The kind of contributions received The kind of contribution might be corrective, adaptive, perfective or preventive, for instance.

4.2 Using the Measures

Naturally, some low-level measures are trivially tied to some objectives. For example, web-site access or number of views for the social media are close to the objective reputation.

Some measures are temporally related. For example, one expects a high rate of downloads before receiving bug reports and feature requests. Similarly, the number of users subscribed to the mailing list could be related to the number of views of social media. Furthermore, one could hypothesize that a set of temporally related measures can be used to analyze the achievement of the same objective.

5 Existing Work

Open source communities have been under heavy research during the last few years. In this paper a few studies and incubation programs are highlighted, and their position is compared to our study.

Works like the book **Producing Open Source Software [8]** contains a practical cookbook style guide for starting and running an OSS community. The book discusses open source development from many aspects ranging from technical infrastructure to political issues of open source. It has a very low-level and detailed view to the phenomenon. For example, everything starts with “choosing a good name” to the project. Due to the fact, that OSS is a fuzzy phenomenon, the book must be taken as an position of the writer. Despite that, it is (to our knowledge) the most throughout review to the subject.

In this paper we have much narrower view angle to the issue. Where [8] has a holistic view to open source, we are concentrating only to an industrial setting. Moreover, our time frame consists only the very beginning of the life cycle of the OS project.

Incubation Programs: Many large industrial open source projects have incubation programs. The programs are used to establish fully functioning well organized subprojects to the metaproject. For example Eclipse platform project and Apache Server Project have both their own incubation programs [3,2].

Apache Incubator[2] is a hatchery for projects that are intended to become actual Apache Foundation's [1] projects. The incubation process is depicted in the Figure 3. First, the process being incubated is established as a candidate, then the candidate process is either accepted or rejected. An accepted project is set to state Podling during which the project is prepared for the review, whose outcome can be termination, continuation or graduation. The last meaning that the project is accepted top ASF project. The Eclipse Project Lifecycle is very much like the Apache Incubator.

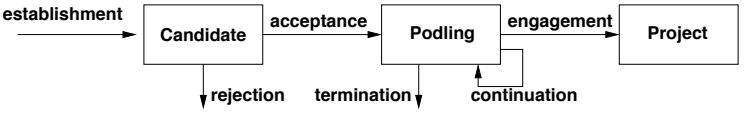

Fig. 3. Apache Incubation process

Referring to the setting in this paper, there is something familiar in the incubation programs. Both are aiming at acceptance of the candidate project. In our case it is accepted (or rejected) by the members of the community, without any formal process. Therefore, the states of the project and the state changes are not clearly visible, but some measurement must be developed.

Open Source Community Building: An inclusive review to OS community building is given in [17]. The licentiate thesis describes a qualitative research of eight successful open source projects. The study is based on interviews with a representative of each project. The interviews go through the life cycles of the projects from the beginning to the current state. The main contribution of the thesis is in describing how to initialize an OS project and how the project is promoted.

The subject of the thesis is pretty close to our work. Both deal with industrial OS project building. The main difference is that we do not base our study to afterward interviews, but participate the actual release process from the beginning. Moreover, we are not passive observers, but we attempt to impact the project building with various ways.

OSS Research: Studying open source projects has been popular lately. However, to our knowledge only one few studies have concentrated on establishing a community [10]. Most of the research, like [5,13,11,16] are different kind of observations and measurements of existing communities. Therefore, our work is a forerunner in the field.

6 Conclusions

This paper studied the problem of releasing industrial closed source software as open source. The problem was formulated as a community building challenge around the

software. We presented an onion-model-based approach which addresses the main stakeholders and concerns involved.

In order to validate our approach, we applied the proposed framework to three industrial cases. The opened products varied from from an open architecture standard with reference implementation to a small software prototype. The target communities varied from closed core and company-based community to open core, volunteer one.

We organized interviews and questionnaires to contextualize the framework and prepare a release plan for each of the cases. Taking into consideration the objectives of the companies, we sketched an evaluation framework that is based on a number of measures in relation to the objectives. The measures are used to observe the achievement of those objectives.

Our results are preliminary in the sense that we are in the early phase of the study. Our early experiences with releasing one of the case studies have been promising. We were able to successfully adapt the framework to the organizational structure of the company, industrial partners and the target community.

In this paper we did not study some issues in detail. For example, laws, cultural issues and the original business motives were out of scope of this study. This is left as a future work. In order to collect actual measurements, companies will be periodically queried for the required data. We plan to organize questionnaires to the industrial partners and new-born open source communities in order to understand better their viewpoint.

References

1. Apache foundation, http://www.apache.org/ (last visited October 2008)
2. Apache incubator, http://incubator.apache.org/ (last visited October 2008)
3. Eclipse incubation phase, http://wiki.eclipse.org/Development_Resources/HOWTO/Incubation_Phase/ (last visited October 2008)
4. Bonaccorsi, A., Rossi, C.: Altruistic individuals, selfish firms? the structure of motivation in open source software. First Monday (1-5) (2004)
5. Capra, E., Wasserman, A.I.: A framework for evaluating managerial style in open source projects. In: Russo, B., Damiani, E., Hissan, S., Lundell, B., Succi, G. (eds.) Proceedings of the fourth International Conference on Open Source Systems, pp. 1–14 (2008)
6. Crowston, K., Howison, J.: The social structure of free and open source software development. First Monday 10(2), 1–100 (2005), http://firstmonday.org/issues/issue10_2/crowston/index.html
7. Fitzgerald, B.: The transformation of open source software. MIS Quarterly 30(3), 587–598 (2006)
8. Fogel, K.: Producing Open Source Software: How to Run a Successful Free Software Project. O'Reilly Media, Inc., Sebastopol (2005), http://www.amazon.fr/exec/obidos/ASIN/0596007590/citeulike04-21
9. Goldman, R., Gabriel, R.: Innovation Happens Elsewhere: How and Why a Company Should Participate in Open Source. Morgan Kaufmann Publishers Inc., San Francisco (2004)
10. Järvensivu, J., Mikkonen, T.: Forging a Community – Not: Experiences on Establishing an Open Source Project. In: Proceedings of the fourth International Conference on Open Source Systems, pp. 15–27 (2008)
11. Kamei, Y., Matsumoto, S., Maeshima, H., Onishi, Y., Ohira, M., itci Matsumoto, K.: Analysis of coordination between developers and users in the apache community. In: Proceedings of the fourth International Conference on Open Source Systems, pp. 81–92 (2008)

12. Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., Ye, Y.: Evolution patterns of open-source software systems and communities. In: Proceedings of the International Workshop on Principles of Software Evolution 2002, pp. 76–85 (2002)
13. Petrinja, E., Sillitti, A., Succi, G.: Overview on trust in large floss communities. In: Proceedings of the fourth International Conference on Open Source Systems, pp. 47–56 (2008)
14. Preece, J.: Online Communities: Designing Usability, Supporting Sociability. Wiley, Chichester (2000)
15. Raymond, E.S.: The Cathedral and the Bazaar. O'Reilly & Associates, Inc., Sebastopol (1999)
16. Studer, M.: Community Structure, Individual Participation and the Social Construction of Merit. In: Feller, J., Fitzgerald, B., Scacchi, W., Sillitti, A. (eds.) Proceedings of the third International Conference on Open Source Systems. IFIP, vol. 234, pp. 161–172. Springer, Heidelberg (2007)
17. Stürmer, M.: Open Source Community Building, licentiate thesis (2005)
18. Välimäki, M.: The Rise of Open Source Licensing: A Challenge to the Use of Intellectual Property in the Software Industry. Turre Publishing (2005)
19. Weiss, M., Moroiu, G., Zhao, P.: Evolution of open source communities. In: Proceedings of the second International Conference on Open Source Systems, pp. 21–32 (2006), doi:10.1007/0-387-34226-5-3