Evolution of Open Source Communities

Michael Weiss, Gabriella Moroiu, and Ping Zhao

Carleton University, School of Computer Science, 1125 Colonel By Dr, Ottawa, Ontario K1S 5B6, Canada weiss@scs.carleton.ca

Abstract. The goal of this paper is to document the evolution of a portfolio of related open source communities over time. As a case study, we explore the subprojects of the Apache project, one of the largest and most visible open source projects. We extract the community structure from the mailing list data, and study how the subcommunities evolve, and are interrelated over time. Our analysis leads us to propose the following hypotheses about the growth of open source communities: (1) communities add new developers by a process of preferential attachment; (2) links between existing communities are also subject to preferential attachment; (3) developers will migrate between communities together with other collaborators; and (4) information flow follows project dependencies. In particular, we are concerned with the underlying factors that motivate the migration between communities, such as information flow, co-worker ties, and project dependencies.

1 Introduction

There is much anecdotal evidence that open source communities grow according to a preferential attachment mechanism [13]. However, there is not much empirical analysis to demonstrate this phenomenon. Most work on open source communities centers on either static aspects of a community (such as its topology at a given time) [9, 14, 15], or describes the evolution of the community in a qualitative manner [16, 8, 4]. The interaction between communities over time (eg the migration of developers) has also not received sufficient attention.

Our goal in this paper is to document the evolution of a portfolio of related open source communities over time. As a case study, we explore the subprojects of the Apache project, both for reasons that this is a highly visible group of open source communities, but also because a wealth of data is being collected on the Apache project site that allows deep insight into the dynamic project structure. In particular, we rely on mining the project mailing lists. Another reason that made this choice conducive was the availability of the Agora [10] tool for extracting information from the Apache project mailing lists.

The paper is structured as follows. Section 2 describes the methodology followed to extract the community structure and various indicators (such as developer rank) from the mailing list data. In Section 3, we show how the various subcommunities of the Apache project evolve, and are interrelated over

Please use the following format when citing this chapter:

Weiss, M., Moroiu, G., and Zhao, P., 2006, in IFIP International Federation for Information Processing, Volume 203, Open Source Systems, eds. Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., (Boston: Springer), pp. 21-32

time. We state our findings in the form of four hypotheses, and provide evidence in their support. Finally, Section 4 presents our concluding remarks.

2 Community Structure

Our goal is to track the evolution of open source communities with time. Communities form around open source projects. They are groups of developers who share a common interest in the project, and who regularly interact with one another to share knowledge, and collaborate in the solution of common problems [16]. Communities are at the core of what is described in [3] as Collaborative Innovation Networks (COINs), highly functional teams characterized by the principles of meritocracy, consistency, and internal transparency. As shown in [16], an open source community co-evolves with its associated project. A project without a community that sustains it is unlikely to survive long-term.

Members of an open source community play different roles, ranging from project leaders (maintainers) and core members (contributors) to active and passive users [13, 14, 16]. Project leaders are often the initiators of the project. They oversee the direction of the project, and make the major development decisions. Core members are members who have made significant contributions to a project over time. Active users comprise occasional developers and users who report bugs, but do not fix them. Passive users are all remaining users who just use the system. Core members can further be subdivided into creators (leaders) communicators (managers), and collaborators [3].

Large open source projects such as GNU, Linux, or Apache comprise many subprojects, not of all of which are strongly connected to one another. They are not associated with a single, homogenous community, but rather an ecology [5] of (sub-)communities is formed around these subprojects. However, they share a common governance (the Apache Foundation, in the case of the Apache project), and often produce artefacts shared among all projects (such as the Jakarta Commons in the Apache project). The idea of an ecology should convey mutual dependencies between many of the projects and cross-project collaboration, but also competition for resources among projects.

Figure 1 shows the current portfolio of projects in the Apache project and their relationships. It depicts the communication patterns between projects, as determined from the project mailing lists. This diagram was generated by an extension of the Agora [10] tool, which reuses its data extraction and core visualization routines, but adds project and module dependency views (based on JDepend [6]), and significant capabilities for pruning by strength of the communication links and filtering by date, as well as statistical analysis.

The structure of a community can be inferred from the interactions between developers on the mailing list of the associate project. We analyze the communication patterns between developers, and order developers by the strength of their communication links. For each developer we tally the number of inbound

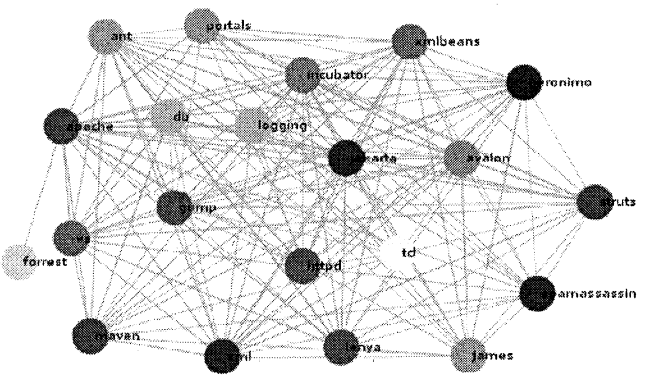


Fig. 1. Portfolio of projects in the Apache project and their relationships

and outbound messages.[1] The project leader is considered the developer with the highest number of inbound messages, as this indicates how frequently this developer is consulted by others. It is, therefore, also a measure of the developer's reputation. The same metric is used in [3] to identify creators, the members who provide the overall vision and guidance for a project.

For the purposes of our analysis, we limit our attention to the group of core developers. According to a previous study of the Apache project [11], most of the contributions are made by the top 15 developers in a project. These are considered the core developers. As noted in [3], a typical core group starts out with 3 to 7 members, and grows to 10 to 15 members, once the community is established. Using the pruning feature of our extended Agora tool, we retrieved the core developers for every subproject of the Apache project. The structure of a community obtained can be visualized as a network of developers.

Fig. 2 shows the community structure of the Httpd subproject based on the messages exchanged over the 01/1999 to 12/1999 time frame.[2] It can be observed that the core group is a nearly fully connected network in which every member communicates directly with every other member. Our database consists of 24 projects and 253 unique core developers. Fig. 3 plots the cumulative number of projects P and developers N for the period of 1997–2004.

[1] The algorithm for extracting topological data from the message set in the Agora tool is is based on the concept of "reply": when a person sends a message in reply to another message, a link is created in the graph. To eliminate noise messages that are not replied to are excluded from the extracted data [10].

[2] The color intensity of the links indicates the strength of a communication link.

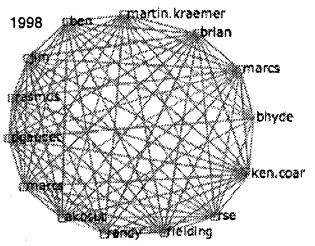


Fig. 2. Communication links between the developers of the Httpd subproject

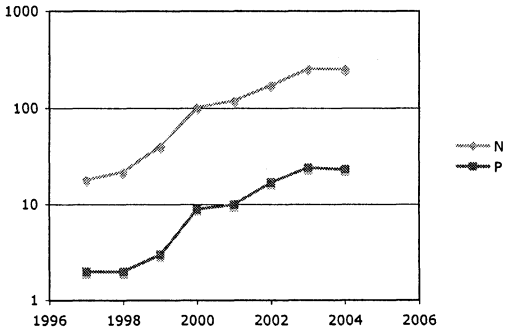


Fig. 3. Cumulative number of projects and developers in the Apache project

3 Tracing Community Evolution

To trace the evolution of a community we took snapshots of its membership at regular intervals. Here, we chose a one year period, but we plan to study the evolution of the Apache communities over smaller time periods in the future. For each period we retrieve the list of core developers ordered by their number of inbound messages, as noted above. The extracted information is captured in a spreadsheet similar to Figure 4 with the nicknames of the core developers for each community and time period. Notably, the top row indicates the project leaders, as inferred from the data. A Perl script translates the spreadsheet data for further processing into a set of Prolog facts. This provide a knowledge base that we can analyze in a flexible manner using the Prolog reasoning engine.

3.1 Growth by Preferential Attachment

Based on this data, we established several hypotheses about the growth of open source communities. Our initial hypothesis that open source communities grow by a process of preferential attachment [9], or selection through professional attention [13] was adopted from the literature. It can be stated as follows:

1997	1998	1999	2000	2001	2002	2003	2004
				donaldp	bloritsch	bloritsch	mcconnell
				bloritsch	peter	mcconnell	niclas
				paul_hamman	paul_hamman	leosimons	bloritsch
				mail	leo.sutic	niclas	leo.sutic
				peter	nicolaken	aok123	dev
				leo.sutic	leosimons	noel	jira
				mcconnell	mcconnell	leo.sutic	cziegeler
				mirceatoma	proyal	alag	aok123
				colus	leif	steve	farra
				charles	craferm	farra	leosimons
				jeff	jeff	nicolaken	noel
				giacomo	noel	cziegeler	develop
				ulim	stefano	holiveira	lsimons
				leif	paulo.gaspar	leosimons	jhawkes
				proyal	cziegeler	paul_hamman	exterminatorx

Fig. 4. Sample of the extracted data (core members of the Avalon subproject)

Hypothesis 1 *The more developers a community has already, the more new developers it will attract (also known as "rich gets richer" phenomenon).*

In support of this hypothesis, we first determine the degree distribution $P(k)$. As shown in Fig. 5, the distribution follows a power law. This indicates that the communication network of the Apache community is scale-free. Such networks contain relatively few highly connected nodes, while the majority of nodes are only connected to few other nodes. This leads to a typical core-periphery structure, as observed for many open source communities.

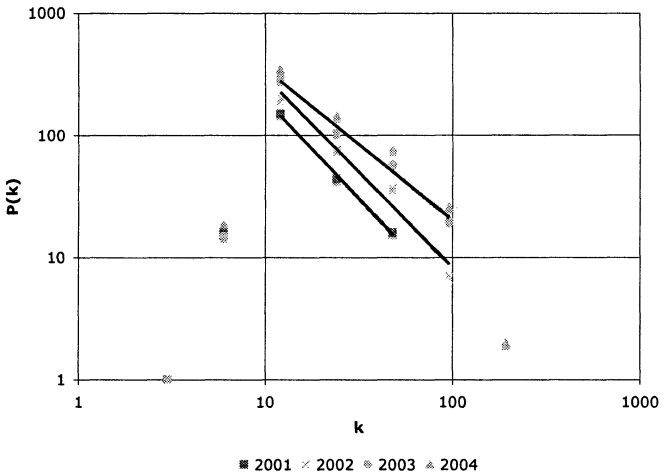


Fig. 5. Developer degree distribution shown with logarithmic binning

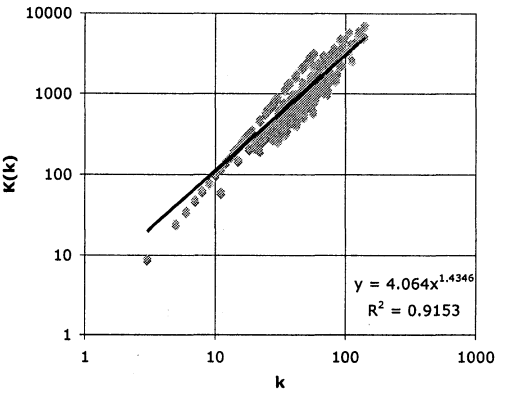


Fig. 6. Cumulative preferential attachment $K(k)$ of new developers

One common mechanism to explain the growth of a scale-free network is preferential attachment [1], as captured by the hypothesis. Preferential attachment implies that, as the network evolves, nodes will link to nodes that already have a large number of links. To verify that the network of the Apache community follows a preferential attachment rule, we determine the probability that a new developer is connected to an existing developer with degree k.

As described in [1], this probability can be estimated by plotting the change in the number of links Δk for an existing developer over the course of one year as a function of k, the number of links at the beginning of each year. Fig. 6 shows the cumulative preferential attachment $K(k)$ of new developers joining the Apache community. If attachment were uniform, $K(k)$ would be expected to be linear. As shown, we find that $K(k)$ is non-linear.

Having established that the growth of the Apache community follows a preferential attachment regime at the developer level, we repeat the analysis at the project level. Instead of estimating the probability of a new developer connecting to an existing developer, we determine the probability of a new developer selecting a given community. In order to show that this probability is proportional to the degree k^{com} of the project community, we determine the change in the number of links for an existing project over the course of one year as a function of the number of links k^{com} at the beginning of each year.

Fig. 7 shows the cumulative preferential attachment $K(k^{com})$ of new developers joining an existing project community. We note that community degree and community size are strongly correlated for higher degrees and larger sizes [12]. Therefore, since the attachment process is preferential with regard to community degree, it is also preferential with regard to community size.

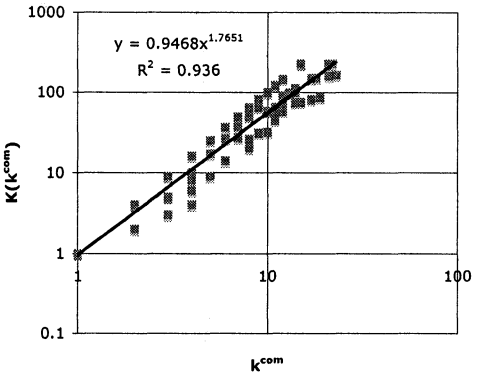


Fig. 7. Cumulative preferential attachment $K(k^{com})$ of new developers

3.2 Interaction and Migration between Projects

As much as the influx of external developers is a key characteristic of open source communities that distinguishes them from other types of networks, it is not the only factor that affects community evolution. As has been noted by [1, 12], the internal interaction between projects also affects the structure and dynamics of a community. Interaction comprises the flow of information, work products, and developers. We will look at each of these aspects below.

Information Flow Information is shared between projects through common developers who act as bridges between the projects. In [4], these developers are considered the "glue that maintains the whole project together, and the chains that contribute to spread information from one part of the project to another".

Hypothesis 2 *The more developers a community shares with other communities, the more developers from other communities will interact with it.*

Fig. 8 shows that the distribution of projects per developer follows a power law. That means that while most developers participate in only few projects, some are active in many projects at the same time. These well-connected developers act as network hubs and facilitate inter-project information flow.

Fig. 9 shows that the number of shared developers grows according to a preferential attachment rule. We obtain this result by plotting the cumulative change $\Delta(k_1^{com}k_2^{com})$ for each pair of projects as a function of $k_1^{com}k_2^{com}$. This estimates the probability that a project with degree k_1^{com} will establish a link with another project with degree k_2^{com}. As shown, the growth is non-linear.

Migration To determine the migration behavior we look at pairs of projects, and test, for each pair P and Q, whether a developer participates in project P is one year and in project Q during the next one, but she is not already a member

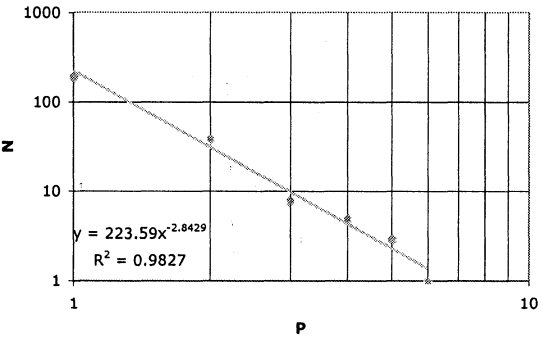


Fig. 8. Distribution of the number of projects per developer

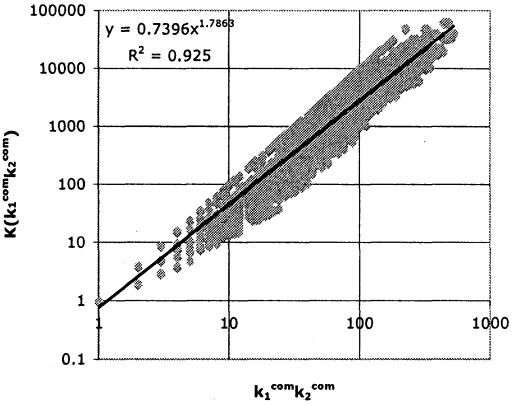


Fig. 9. Cumulative internal preferential attachment $K(k_1 k_2)$ between projects

of project Q in the current year.[3] Fig. 10 shows the developer migration from 2003 to 2004. Each row contains the number of developers migrating from a given project to any of the other projects during the following year. Note that "pool" is not a project, but indicates the influx of new core developers.

Many of these developers migrate to new projects, of which they form the core to which new developers attach themselves. As projects are spun off from existing projects, developers tend to migrate with community members they closely associate with. We should expect the effect to be most pronounced, if the leader of one project moves on to a new project: this would create an even stronger pull for other core developers to join the new project. Thus, we surmise that developer *reputation* also plays a critical role in migration decisions.

[3] This is an example of a rule that we can easily model and evaluate in Prolog. However, space does not allow us to describe the details of this modeling step.

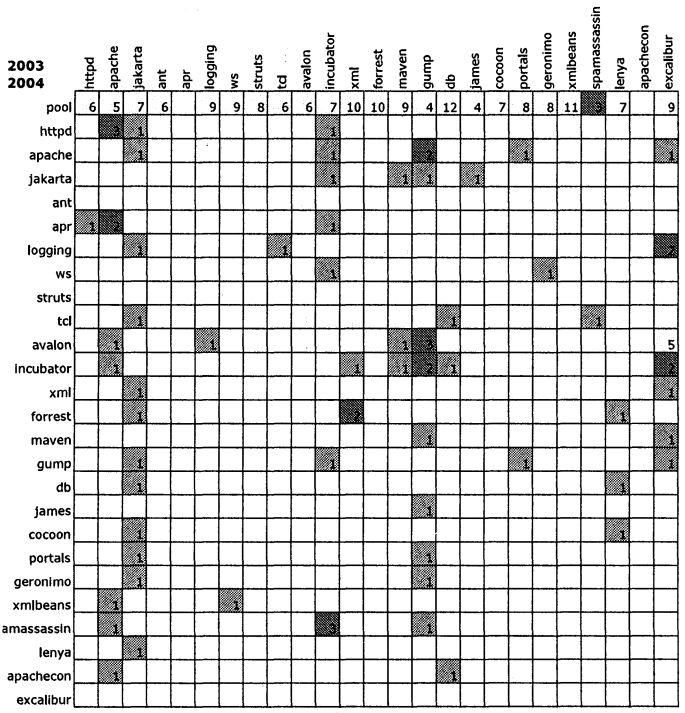


Fig. 10. Migration between projects from 2003-2004

Hypothesis 3 *Developers will migrate between communities with their collaborators, that is, other developers with which they have strong ties.*

Fig. 11 plots the distribution $P(s)$ of group size s. It can be seen to observe a power law. This supports the hypothesis. While many developers will migrate in small groups, some well-connected developers will move in large groups, which provide the support for a new project. Our data supports that most new projects include at least one large group migrated from another project.

As an example, consider the migration into the Excalibur project shown in Fig. 12. The Excalibur project receives its main contribution from the Avalon project. A drill-down into the underlying data reveals that the current leader of the Avalon project (bloritsch), as well as the future leader of the Excalibur project (leosimons) are among those developers. The leader of the Avalon project brings with him four co-workers from that project.

Project Dependencies Sharing of work products takes the form of shared modules. It can be observed in different ways, eg from the developer attributions in a code repository as in [4], or from an analysis of the import statements in the source code. Our extensions to Agora includes a module dependency view, which

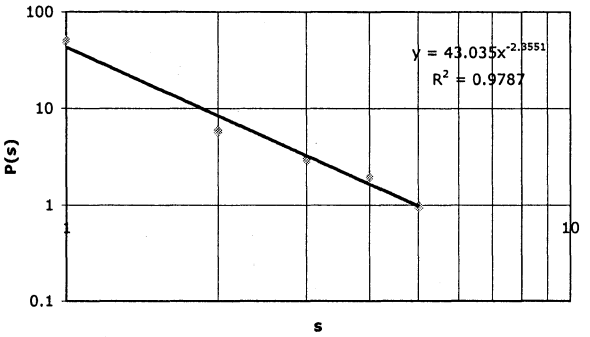


Fig. 11. Distribution of migration group size (transition from 2003-2004)

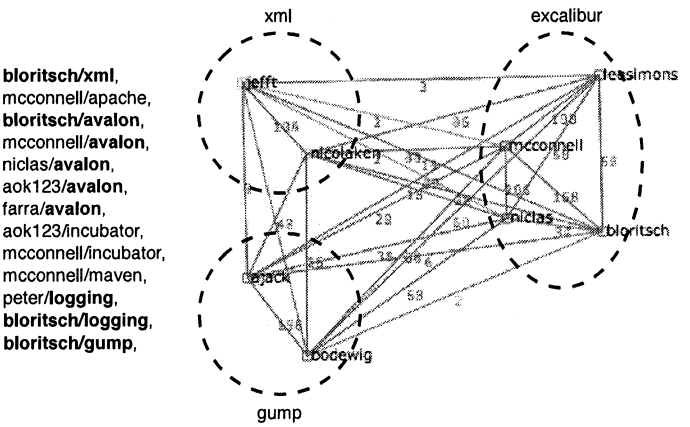
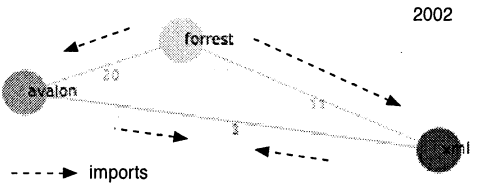


Fig. 12. Migration to the Excalibur project between 2003 and 2004

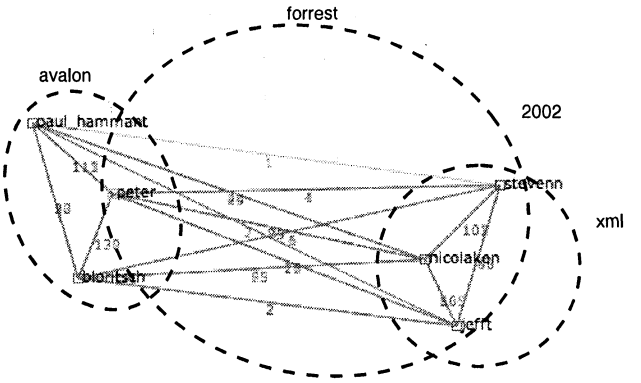
presents information extracted from the project source code using JDepend [6] as a graph. Links in the graph indicate module dependencies.

Hypothesis 4 *Information flow follows project dependencies.*

While we have not yet extracted dependency information on all subprojects in the Apache project, we have analyzed project dependencies for specific cases, as triggered by observations made during our analysis of information flow or developer migration. As an example of the kind of analysis, we can perform with Agora, Fig. 13 shows the dependencies between the Agora, Forrest, and XML projects (top), and corresponding information flow (bottom). It can be seen that there is one core developer bridging the Avalon and Forrest communities, and that the Forrest and XML projects share three core developers.



(a) Project dependencies



(b) Information flow between projects

Fig. 13. Project dependencies between the Agora, Forrest, and XML projects in 2002

4 Conclusion

In this paper, we stated a set of hypotheses about the evolution of open source communities. As a first step of the empirical validation of these hypotheses, we presented our initial results exploring the communities formed around the various subprojects of the Apache project. To this end we extended a tool (Agora) developed by a member of the Apache project with project and module dependency views, and pruning and date filtering capabilities, as well as statistics.

We then extracted information about the core developers of each community over an eight year time period (1997–2004). This data allowed us to explore the hypotheses in some detail through various cases, where we documented the migration behavior of developers between selected project communities. We also built an exploratory tool in Prolog for rapidly modeling and testing new hypotheses about the extracted data. We were able to identify different factors that underlie the preferential attachment mechanism of community evolution, including information flow, co-worker ties, and project dependencies.

References

1. Barabasi A, Jeong H, et al (2002) Evolution of the Social Network of Scientific Collaborations, Physica A 311, 590-614
2. Feller J, Fitzgerald B, Hissam S, Lakhani K (2002) Perspectives on Free and Open Source Software, MIT Press
3. Gloor P (2006) Swarm Creatitity, Oxford University Press
4. Gonzalez-Barahona J, Lopez L, Robles G (2004) Community Structure of Modules in the Apache Project, Workshop on Open Source Software Engineering
5. Healy K, Schussman A (2003) The Ecology of Open Source Development, Unpublished, `www.kieranhealy.org/files/drafts/oss-activity.pdf`
6. JDepend (2006) Project, `www.clarkware.com/software/JDepend.html`, last accessed in Jan 2006
7. Koch S (2005) Free/Open Source Software Development, Idea Publishing
8. Koch S (2005) Evolution of Open Source Software Systems – A Large-Scale Investigation, International Conf on Open Source Systems, 148-153
9. Madey G, Freeh V, Tynan R (2005) Modeling the F/OSS Community: A Quantitative Investigation, in [7], 203–220
10. Mazzocchi S (2006), Apache Agora 1.2, `people.apache.org/~stefano/agora/`, last accessed in Jan 2006
11. Mockus A, Fielding R, Hersleb J (2005) Two Case Studies of Open Source Software Development: Apache and Mozilla, in [2], 163–209
12. Pollner P, Palla G, Viczek T (2006) Preferential Attachment of Communities: The Same Principle, But at a Higher Level, Europhysics Letters, 73 (3), 478–484
13. van Wendel R, de Bruijn J, van Eeten M (2003) Protecting the Virtual Commons, Information Technology & Law Series, T.M.C. Asser Press, 44–50
14. Xu J, Madey G (2004) Exploration of the Open Source Software Community, NAACOSOS Conf, no page numbers
15. Xu J, Gao Y, et al (2005) A Topological Analysis of the Open Source Software Development Community, Hawaii International Conf on System Sciences, 1–10
16. Ye Y, Nakakoji K, et al, The Co-Evolution of Systems and Communities in Free and Open Source Software Development, in [7], 59–82