Open Source Community Processes: Implications on Micro and Macro Level

Auswirkungen von Prozessen in Open Source Gemeinschaften auf Mikro- und Makro-Ebene

Stefan Koch*, Bogazici University, Istanbul, Turkey

* Correspondence author: stefan.koch@boun.edu.tr

Summary In this paper, we will discuss the existence and form of open source community processes, whether they differ between projects, and whether they have any implications on both micro and macro level. We will take a look at how different processes can impact the work and the resulting quality within projects, as well as on macro or project level outcomes like success or efficiency. We will also present as well as validate a research model to explain process adoption and implications. ►►► **Zusammenfassung** Dieser Artikel thematisiert die Prozesse in Open Source Gemeinschaften. Wir diskutieren sowohl Existenz wie auch Form solcher Prozesse, und fokusieren auf die Auswirkungen unterschiedlicher Prozesse auf sowohl Qualität und Zusammenarbeit innerhalb von Projekten, aber auch auf Attribute auf Projekt-Ebene wie den Erfolg. Weiter präsentieren wir ein Modell zur Erklärung von Prozessdesign sowie -auswirkungen.

Keywords ACM CCS → Software and its engineering → Software creation and management → Collaboration in software development → Open source model; ACM CCS → Software and its engineering → Software creation and management → Software development process management; ACM CCS → Software and its engineering → Software notations and tools → Software configuration management and version control systems ►►► **Schlagwörter** Open Source, Softwareentwicklung, Softwarequalität, Verteilte Zusammenarbeit, Aufwand, Effizienz

1 Introduction

Open source software has become more and more important, with this now stretching beyond the mere use of well-known projects in both private and commercial settings [9]. Open source software is of special interest also due to its development processes and organization of work. In many ways it is seen as constituting a new production mode, in which people are no longer collocated, and self-organization is prevalent. One of the first opinion pieces on this model was written by Eric S. Raymond, 'The Cathedral and the Bazaar', in which he contrasts the traditional type of software development of a few people planning a cathedral in splendid isolation with the new collaborative bazaar form of open source software development [31]. In his theoretical description, a large number of developer-turned users come together without monetary compensation to cooperate under a model of rigorous peer-review and take advantage of parallel debugging that leads to innovation and rapid advancement in developing and evolving software. In order to allow for this to happen and to minimize duplicated work, the source code needs to be accessible, and new versions need to be released in short cycles.

In this paper, we will discuss in Sect. 2 open source community processes resulting from self-organizing projects, and whether they differ between projects in ma-

turity or division of labour. We will take a look at whether these processes have any implications on both the micro level like modules, as well as the macro or project level, e. g. how different processes can impact the work and the resulting quality within projects, as well as outcomes like success or efficiency in Sect. 3. We will adress the question of which characteristics lead to process adoption, and its implications by presenting as well as validating a research model in Sect. 3.3, and conclude with directions for future research in Sect. 4.

2 Open Source Community Processes and Organization of Work

Quite often open source software development is understood as a new production mode, lacking collocation of members as well as any centralized management, or defined processes in any way, based on self-organization [7]. While there is one seminal description of the bazaar style of development by Raymond [31], it should be noted that reality has been found in many cases to differ from this theoretical description, and significantly between projects [33]. The distribution of effort and output between developers has been found to be heavily skewed, and there is both evidence of clearly defined processes, for example strict release processes in several projects [16], as well as a considerable level of commercial involvement [32]. In addition to that, emergent but not necessarily written, modeled and defined governance structures, roles and processes in general do exist. In addition, most projects fail to attract a community of developers and do not achieve progress or success in any way. We will try to establish a model for process adoption and implications in this paper.

O'Mahony and Ferraro [29] specifically focused on this emergence of governance, finding that members develop a shared basis of formal authority but limit it with democratic mechanisms that enabled experimentation with shifting conceptions of authority over time. Barcellini et al. [2] find community consensus as well as implicit rules to govern design dynamics, as well as specific participants ("top hierarchy") active in framing. In a second, related study [1], the authors find several key participants acting as boundary spanners between user and developer communities. They therefore argue that OSS design may be considered as a form of "role emerging design", i. e. design organized and pushed through emerging roles and through a balance between these roles, with the communities providing a suitable socio-technical environment to enable such role emergence. Within open source projects, different levels and roles generally exist. Von Krogh et al. [37] explicitly focus on joining projects, and found that following certain joining scripts, helped by specialization and volunteering feature gifts, leads to a higher chance of gaining access to the developer community.

Several ways have been discussed to describe different open source development processes, e. g. Crowston et al. [6] operationalize a process characteristic based on the speed of bug fixing, Michlmayr used a construct of process maturity [27], while also concentration indices have been used to characterize development forms [23]. We find that there is considerable variance in the practices actually employed, as well as the technical infrastructure adopted to enforce them, or in turn prescribing and shaping them. It has been hypothesized that the advent of the Internet and especially the coordination and communications tools are at least a precondition for this development. On a conceptual level, Hemetsberger and Reinhardt [14] draw on the concept of co-configuration, which is is a participatory model that integrates users as active subjects in the shaping and reshaping of products and who eventually become experts themselves. They use the term of coat-tailing work systems that tie everyday actions to the overall activity of the group, and underscore the importance of technological, cultural and mental artifacts to construct such a system.

Numerous studies of projects and communities have proposed metrics like commits, the number of distinct programmers involved in a file or project, or the Gini concentration coefficient to study open source work practices. One of the most consistent results is a heavily skewed distribution of effort between participants. For example, Mockus et al. [28] have shown that the top 15 of nearly 400 programmers in the Apache project added 88% of the total lines-of-code. In the GNOME project, the top 15 out of 301 programmers were only responsible for 48%, while the top 52 persons were necessary to reach 80% [24], with clustering hinting at the existence of a still smaller group of 11 programmers within this larger group. A similar distribution for the lines-of-code contributed to the project was found in a community of Linux kernel developers [15], or in the Orbiten Free Software survey [11], where the first decile of programmers was responsible for 72% of the total code.

A second major result regarding organization of work is a low number of people working together on file level. For example, one study found that only 12.2% of the files have more than three distinct authors [23]. Most of the files have one (24.0%) or two (56.1%) programmers and only 3% have more than five distinct authors, in accordance with other studies on file or project level [11; 18; 28]. We find that these results are in line with seeing open source development as a more component-oriented approach, as compared to agile methods which are more feature-oriented. Similar distribution can also be found on project level in large scale studies: For example, previous research based on several thousand projects found a vast majority of projects having only a very small number of programmers (67.5% have only 1 programmer). Only 1.3% had more than 10 programmers [18].

Another aspect that cannot be underestimated with regard to the implications for existence and form of community processes is the increased commercial interest. This has also lead to changes in many projects, which now

include contributors who get paid for their contributions. This can have repercussions on motivation and participation [30], and is also reflected in several surveys: For example, Lakhani and Wolf [25] found that 13% of respondents received direct payments, and 38% spent work hours on open source development with their supervisor being aware of the fact. Ghosh [10] reports a group of 31.4% motivated by monetary or career (mostly for signaling competence) concerns. Hars and Ou [12] found a share of 16% being directly paid, Hertel et al. [15] report 20% of contributors receiving a salary for this work on a regular basis in a survey of Linux kernel developers.

3 Implications on Micro and Macro Level

3.1 Implications for Quality

The quality resulting from open source development and its assurance is a major concern, and a hugely debated topic. We will therefore highlight a few results which link elements of open source community processes to the quality achieved, as measured by diverse metrics from software engineering like McCabe's cyclomatic complexity or Chidamber and Kemerer's object-oriented metrics, which try to capture different aspects related to coding and design quality, e. g. complexity, which in turn have been shown to have a major impact on aspects like maintainability.

First focusing on the micro-level within projects, one study found that a high number of programmers and commits, as well as a high concentration is associated with problems in quality on class level, mostly to violations of size and design guidelines, thus being related to higher bug counts as well as problems in maintenance [23]. If the architecture is not modular enough, a high concentration might show up as a result of this, as it can preclude more diverse participation. The other explanation is that classes that are programmed and/or maintained by a small core team are more complex due to the fact that these programmers 'know' their own code and do not see the need for splitting large and complex methods.

On project level, there is a distinct difference [23]: Those projects with high overall quality ranking have more authors and commits, but a smaller concentration than those ranking poorly. Thus, on class level a negative impact of more programmers was found, while on project level a positive effect. This underlines a central statement of open source software development, that more people within a project will lead to higher quality and more progress. If possible, these resources should, from the viewpoint of product quality, be structured in small teams. Ideally, on both levels, the effort is not concentrated on too few of the relevant participants. Underlining these results, MacCormack et al. [26] find that a product's design mirrors the organization developing it, in that a product developed by a distributed team such as the Linux kernel was more modular compared to Mozilla developed by a collocated team. Alternatively, the design also reflects purposeful choices made by the developers based on contextual challenges, in that Mozilla was successfully redesigned for higher modularity at a later stage.

Finally, the involvement of developers with a commercial interest seems to affect the way of work, for example a large amount of commercial interest has led to a governance structure which puts great value on control and stability by requiring technical improvement proposals and an associated process for major changes in the OpenACS project [8]. Packages dominated by commercial background tend to include less developers overall and less volunteers, and also tend to be changed less often and by the same group of people [8]. Again, this is quite contrary to the open source development model, and could create serious problems related to quality as well as long-term maintainability.

3.2 Implications for Project Effort and Efficiency

When we try to estimate the effort that is expended in open source communities, we find that it seems to be much less than in a comparable commercial setting [20]. This difference might be due to several reasons. First, open source development organization might constitute a more efficient way of producing software, mostly due to self-selection outperforming management intervention. Participants might be able to determine more accurately whether and where they are able to work productively on the project overall, or on particular tasks. In addition, overhead costs for planning, budegting, controlling and other related tasks are very much reduced. A second explanation might be that the difference is caused by non-programmer participation, i. e. people participating by discussing on mailing lists, reporting bugs, maintaining web sites and the like. This could be interpreted as supporting the idea of open source communities following processes similar to the 'chief programmer team organization' or 'surgical team' [3], where system development is divided into tasks each handled by a chief programmer who has responsibility for the most part of the actual design and coding, supported by a larger number of other specialists such as documentation writers or testers.

For focusing on the implications on efficiency and success of community processes, one major problem is that conceptualizing these aspects is difficult for open source communities. Many researchers have worked on conceptualizing the success of such projects [5; 6; 35], suggesting a huge number of indicators, using, for example search engine results, or measures like number of downloads, or even subjective answers to surveys. The first element to be explored for any connection to these aspects naturally is the potentially large number of participants. Following the reasoning of Brooks [3], an increased number of people working together will decrease productivity due to communication costs. Interestingly, this effect has not turned up in prior studies [18; 19; 34], respectively only within the group of core developers [4]. This leads to

the conclusion that Brooks's Law seems not to apply to open source software development, maybe because of very strict modularization, which increases possible division of labor while reducing the need for communication. Also the low number of programmers working together on single files can be taken as a hint for this. Similarly, a very inequal distribution of effort is not connected to efficiency [21], so this aspect of community processes does not seem to incur a penalty.

Closely linked to efficiency in a project is its rate of progress and growth. For open source software systems, several works have explicitly dealt with this topic. Most of them have found super-linear growth, which contradicts the prior theory of software evolution, e. g. for Linux or KDE. With a large sample of projects, one study came to the conclusion that while in the mean the growth rate is linear or decreasing over time according to the laws of software evolution, a significant percentage of projects is able to sustain super-linear growth [19]. There is a positive relationship between the size of a project, the number of participants and the inequality in the distribution of work within the development team with the presence of super-linear growth patterns.

Finally, the infrastructure employed for communication and coordination naturally shapes the processes in a project. For example, Michlmayr [27] has used a sample of projects to uncover whether the process maturity, based on version control, mailing lists and testing strategies, has had any influence on the success of open source projects, and could confirm this. Within SourceForge the impact of adoption of different tools offered by as well as tool diversity on project efficiency has been analyzed with surprising results [22]: In a data set of successful projects, actually negative influences of tool adoption were found, while the results were more positive in a random data set. Some projects, especially larger ones, might be using tools hosted in other places. Another explanation is that tools for communicating with users and potential co-developers can become more of a hindrance in successful projects, as they could increase the load to a degree that it detracts attention and time. In addition, these projects in general have a higher number of developers, so maybe projects with problems in communication and coordination due to team size adopt tools to a higher degree, which can not completely solve the problem after a certain point. In addition, tool adoption might increase user involvement, which in turn could influence project design and fit with user expectations. This could also lead to an actual reduction in planned functionality, if it does not add value to the user base, as well as optimization in code. Therefore, the size of the software, which is one output aspect considered, could actually be reduced, leading to a perceived lower efficiency.

3.3 Model for Process Adoption and Implications

Finally, we will try to provide a model explaining process adoption and its implications. We propose that the adoption of processes will be mostly driven by the size of the community, as well as their diversity. We also add the age of a project as an explanatory variable for size and diversity of the community, as these will generally be time-dependent aspects. If the members of the community are more widely distributed with regard to background, but also language or location, the need for processes to govern and facilitate interactions becomes much greater. We also presume that the adoption of tools like source code versioning or bug tracking is closely related to this adoption of more formalized processes. On the one hand, formalization of processes will lead to tool adoption, but also tool adoption prescribes and necessitates the adoption of some form of process. For example, bug tracking tools generally assume or prescribe some form of bug handling process. The main hypothesis then is that the adoption of processes and tools will be related to project success. We explicitly add a direct impact of diversity and size of community on the success as well to see both direct and indirect effects. Increased diversity in a team, besides some negative effects covered above, can have positive effects due to increased creativity, problem-solving ability, marketing to diverse groups, or effectiveness [13; 30].

For this research we chose to use data available from SourceForge.net, the largest hosting platform for free and open source projects. The necessary data was collected for each project in the data set from the respective homepage (with some exceptions, e. g. pertaining to the source code). We selected the 30 most most downloaded projects, but complemented with 100 randomly selected projects. It has to be noted that using data from SourceForge.net raises several issues of validity, for which Howison and Crowston [17] give an overview. For the constructs used above, we will employ the following operationalizations: The age of the project is measured in years since project registration. For the size of the community, we use the number of developers of the project. For the diversity, we aggregate three different aspects: If the license is something else then GNU-style, this is going to lead to greater diversity within the development community due to the public perception of this license as well as greater chances for commercial involvement, as is if the intended audience is not limited to developers and/or system administrators. Finally we check whether the number of translations is more than two, and aggregate all three aspects from binary variables by summing up. For process and tool adoption, we check whether source code control system, tracker, mailing list, forum, tasks and wiki are used. The success is operationalized by the number of downloads as in many previous studies, as this measure that is presumed among other factors to be dependent on aspects like functionality, size and quality. For a detailed discussion of the methodology and resulting threats to validity, especially construct validity, the reader is refered to [22].

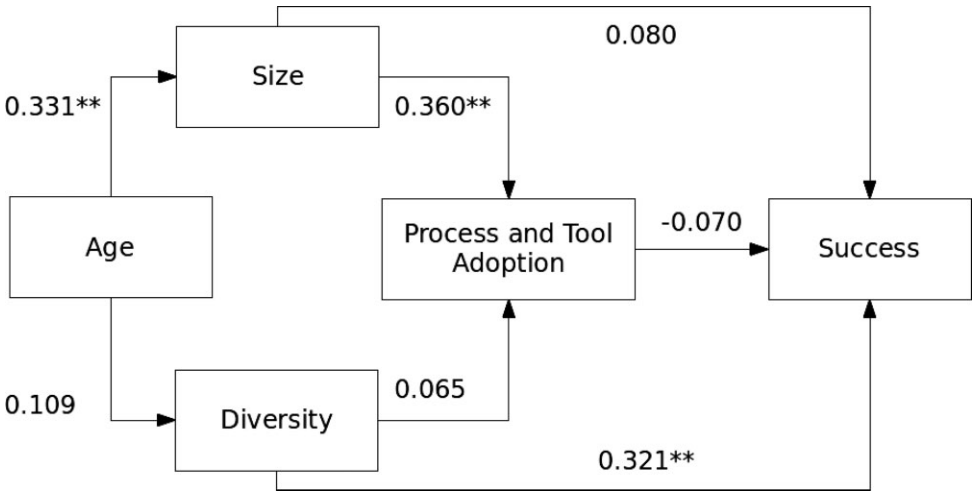

Figure 1 Research model with results from structural equation modeling (showing standardized coefficients, ** $p < 0.01$).

We tested the proposed model using structural equation modeling (SEM). The results are shown in Fig. 1. As can be seen, not all proposed relationships hold. Age of a project does have a significant effect on team size, but not diversity. Process and tool adoption is driven only by size, not diversity. A possible explanation might be that open source teams are locally distributed even without additional effects as covered by our diversity construct, which leads to a need for tools and processes once a certain size is passed. For success, only diversity showed a clear impact. As we operationalized this using the number of downloads, it seems that the appeal to a wider range of users is mostly driven by the creativity of the group, not the size or processes.

4 Conclusion and Future Research

We have tried to highlight some aspects of open source community processes and how they impact aspects of the projects, both on the micro level like modules, as well as the macro or project level. We have shown that the processes within open source communities can differ significantly, and that different forms of organization within projects, e. g. different levels of concentration on a core team, can have significant impacts. That means an assessment of these processes should be part of a project evaluation, e. g. for adoption or reuse in a commercial context. Several assessment schemes like Open Business Readiness Rating (OpenBRR), Open Source Maturity Model, QSOS, or OpenBQR have been developed, and some contain aspects of this [36]. Finally, we have tried to develop and validate a novel model for both antecedents of process adoption as well as impact on success. We found that it is mostly the size of the community driving adoption, but that the main determinant of success as measured by downloads is team diversity, which is a new construct proposed for the first time. This draws further attention to issues of attraction and team composition, and would call for future work on these topics within open source communities for example related to the way projects construct their norms and cultures, and whether this fosters diversity.

References

[1] Barcellini, F., Detienne, F., and Burkhardt, J. M. (2008). User and developer mediation in an Open Source Software community: Boundary spanning through cross participation in online discussions. International Journal of Human-Computer Studies, 66, pp. 558–570.

[2] Barcellini, F., Detienne, F., Burkhardt, J. M., and Sack, W. (2008). A socio-cognitive analysis of online design discussions in an Open Source Software community. Interacting with Computers, 20, pp. 141–165.

[3] Brooks Jr., F. P. (1995). The Mythical Man-Month: Essays on Software Engineering. Anniversary ed., Addison-Wesley, Reading, MA.

[4] Capiluppi, A. and Adams, P. J. (2009). Reassessing Brooks' law for the free software community. In Open Source Ecosystems: Diverse Communities Interacting (pp. 274–283). Springer Berlin Heidelberg.

[5] Crowston, K., Annabi, H., and Howison, J. (2003). Defining Open Source Software Project Success. In Proceedings of ICIS 2003, Seattle, WA.

[6] Crowston, K., Howison, J., and Annabi, H. (2006). Information systems success in free and open source software development: theory and measures. Software Process: Improvement and Practice, 11(2), 123–148.

[7] Crowston, K., Li, Q., Wei, K., Eseryel, Y., and Howison, J. (2007). Self-organization of teams for free/libre open source software development. Information and Software Technology, 49, pp. 564–575.

[8] Demetriou, N., Koch, S., and Neumann, G. (2007). The Development of the OpenACS Community. In Lytras, M. and Naeve, A. (eds.) Open Source for Knowledge and Learning Management: Strategies Beyond Tools, Hershey, PA: Idea Group.

[9] Fitzgerald, B. (2006). The Transformation of Open Source Software. MIS Quarterly, 30(3), pp. 587–598.

[10] Ghosh, R. A. (2005). Understanding free software developers: Findings from the FLOSS study. In Feller, J., Fitzgerald, B., Hissam, S. A., and Lakhani, K. R., editors, Perspectives on Free and Open Source Software, pages 23–46. MIT Press, Cambridge, MA.

[11] Ghosh, R. A. and Prakash, V. V. (2000). The Orbiten Free Software Survey. First Monday, 5(7).

[12] Hars, A. and Ou, S. (2001). Working for free? – Motivations for participating in Open Source projects. In Proceedings of the 34th Hawaii International Conference on System Sciences, Hawaii.

[13] Harvey, C. P. and Allard, M. J. (2012). Understanding and Managing Diversity (5th ed.). New Jersey: Pearson Education, Inc.
[14] Hemetsberger, A. and Reinhardt, C. (2009). Collective Development in Open-Source Communities: An Activity Theoretical Perspective on Successful Online Collaboration. Organization Studies, 30(9), pp. 987–1008.
[15] Hertel, G., Niedner, S., and Hermann, S. (2003). Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel. Research Policy, 32(7), pp. 1159–1177.
[16] Holck, J. and Jorgensen, N. (2004). Do not check in on red: Control meets anarchy in two open source projects. In Koch, S., editor, Free/Open Source Software Development, pages 1–26. Idea Group Publishing, Hershey, PA.
[17] Howison, J. and Crowston, K. (2004). The perils and pitfalls of mining SourceForge. In: Proc. of the International Workshop on Mining Software Repositories. Edinburgh, Scotland, pp. 7–11.
[18] Koch, S. (2004). Profiling an open source project ecology and its programmers. Electronic Markets, 14(2), pp. 77–88.
[19] Koch, S. (2007). Software Evolution in Open Source Projects – A Large-Scale Investigation. Journal of Software Maintenance and Evolution, 19(6), pp. 361–382.
[20] Koch, S. (2008). Effort Modeling and Programmer Participation in Open Source Software Projects. Information Economics and Policy, 20(4), pp. 345–355.
[21] Koch, S. (2008). Measuring the Efficiency of Free and Open Source Software Projects Using Data Envelopment Analysis. In Sowe, S. K., Stamelos, I., and Samoladas, I. (eds.): Emerging Free and Open Source Software Practices, pp. 25–44, Hershey, PA: IGI Publishing.
[22] Koch, S. (2009). Exploring the Effects of SourceForge.net Coordination and Communication Tools on the Efficiency of Open Source Projects using Data Envelopment Analysis. Empirical Software Engineering, 14(4), pp. 397–417.
[23] Koch, S. and Neumann, C. (2008). Exploring the Effects of Process Characteristics on Product Quality in Open Source Software Development. Journal of Database Management, 19(2), pp. 31–57.
[24] Koch, S. and Schneider, G. (2002). Effort, Cooperation and Coordination in an Open Source Software Project: Gnome. Information Systems Journal, 12(1), pp. 27–42.
[25] Lakhani, K. R. and Wolf, R. G. (2005). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. In Feller, J., Fitzgerald, B., Hissam, S. A., and Lakhani, K. R., editors, Perspectives on Free and Open Source Software, pages 3–22. MIT Press, Cambridge, MA.
[26] MacCormack, A., Rusnak, J., and Baldwin, C. Y. (2006). Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. Management Science, 52(7), pp. 1015–1030.
[27] Michlmayr, M. (2005). Software Process Maturity and the Success of Free Software Projects. In Zielinski, K. and Szmuc, T. (eds.): Software Engineering: Evolution and Emerging Technologies, pp. 3–14, Amsterdam, The Netherlands: IOS Press.
[28] Mockus, A., Fielding, R., and Herbsleb, J. (2002). Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11(3), pp. 309–346.
[29] O'Mahony, S. and Ferraro, F. (2007). The Emergence of Governance in an Open Source Community. Academy of Management Journal, 50(5), pp. 1079–1106.
[30] Pelled, L. H. Eisenhardt, K. M., and Xin, K. R. (1999). Exploring the black box: An analysis of work group diversity, conflict, and performance. Administrative Science Quarterly, 11, pp. 1–28.
[31] Raymond, E. S. (1999). The Cathedral and the Bazaar. Cambridge, Massachusetts: O'Reilly & Associates.
[32] Roberts, J. A., Hann, I.-H., and Slaughter, S. A. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. Management Science, 52(7), pp. 984–999.
[33] Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K. (2006). Understanding Free/Open Source Software Development Processes. Software Process: Improvement and Practice, 11(2), pp. 95–105.
[34] Schweik, C. M., English, R. C., Kitsing, M., and Haire, S. (2008). Brooks' versus Linus' law: an empirical test of open source projects. In Proceedings of the 2008 international conference on Digital government research (pp. 423–424). Digital Government Society of North America.
[35] Stewart, K. J., Ammeter, A. P., and Maruping, L. M. (2006). Impacts of Licence Choice and Organisational Sponsorship on User Interest and Development Activity in Open Source Software Projects. Information Systems Research, 17(2), pp. 126–144.
[36] Stol, K. J. and Babar, M. A. (2010). A comparison framework for open source software evaluation methods. In Open Source Software: New Horizons (pp. 389–394). Springer Berlin Heidelberg.
[37] Von Krogh, G., Spaeth, S., and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. Research Policy, 32(7), pp. 1217–1241.

Received: March 23, 2013

Prof. Dr. Stefan Koch is Professor and Chair at Bogazici University. His research interests include user innovation, the open source development model, the evaluation of costs and benefits from information systems, and ERP systems. He has published over 20 papers in peer-reviewed journals, and also serves as Editor-in-Chief of the International Journal on Open Source Software & Processes.

Address: Department of Management, Bogazici University, 34342 Bebek, Istanbul, Turkey,
e-mail: stefan.koch@boun.edu.tr