How are Open Source Practices Possible within a Medical Diagnostics Company? Developing and Testing a Maturity Model of Inner Source Implementation

Remo Eckert
University of Bern, Institute of Information Systems
Bern, Switzerland
remo.eckert@iwi.unibe.ch

Sathya Kay Meyer
University of Bern, Institute of Information Systems
Bern, Switzerland
sathya.meyer@students.unibe.ch

Matthias Stuermer
University of Bern, Institute of Information Systems
Bern, Switzerland
matthias.stuermer@iwi.unibe.ch

ABSTRACT
Open Source Software (OSS) development has seen a considerable increase in attention over the last few years. The success of various OSS projects, such as Linux and Apache, is now widely recognized. Many organizations have shown interest not only in using OSS, but also in applying the underlying collaborative practices within their internal software development activities; this phenomenon is known as Inner Source. By combining best practices of OSS development from the current Inner Source literature, we develop a new model that allows us to rate an organization's maturity level regarding the adoption of Inner Source. By testing our model within a medical diagnostics corporation, we present various insights on Inner Source efforts and how Inner Source can improve software development.

Author Keywords
Inner Source; Open Source Software; Maturity Model; Software Development.

ACM Classification Keywords
K.6.3 [Software Management]: Software development.

INTRODUCTION
The success of OSS projects has generated considerable interest over the last few years. There are various examples, such as Linux, Apache and several others that bear testament to this [3]. As a consequence, OSS is also becoming more popular - mainstream even [5]. Organizations hope to leverage the numerous advantages of OSS [14], such as the potential to decrease cost and risk throughout the product lifecycle and minimizing vendor lock-in [32]. However, the way software is developed in an OSS community differs from other forms of software development. Organizations have adopted OSS development practices to improve their internal software development process and thereby gaining efficiency and effectiveness without revealing own source code. This phenomenon is known as *Inner Source*, a term first mentioned in the scientific context by Dinkelacker et al. [3]. Inner Source promises many advantages, including organization-wide and immediate access to all project artifacts and source code; a greater number of releases and shorter time to market [22, 28]; reusable software components [22]; as a form of intra-organizational open innovation [17] and peer-review of contributions through organization-wide scrutiny [8, 15, 21]. Moreover, Riehle et al. [20] list several expected benefits when implementing Inner Source. However, Inner Source also brings a number of challenges. Dinkelacker et al. [3] distinguish between organizational and technology infrastructure challenges faced by organizations implementing Inner Source. Most organizations have a hierarchical organizational structure, which complicates the process of sharing code, because it can be difficult to merge together different product road-maps and time-lines. Security aspects, such as who can access the source code and the migration to new tools and support of those tools need to be targeted [3]. As Inner Source means openness and transparency, employees may understand this shift as an attempt by management to gain more control. They may be concerned that sharing their source code within the organization may uncover errors and bad code quality. Thus, a cultural change is needed to motivate engineers working more open and collaborative [18]. Each implementation of Inner Source depends on the specific organizational context [6, 27]. There have been numerous case studies and observations of firms implementing Inner Source and the challenges they faced when doing so. In their study, Stol et al. [28] list several success stories from case studies conducted at Alcatel-Lucent, HP, Nokia, Philips and SAP. These studies all list various challenges faced by the companies during the implementation.

While the existing literature and case studies on Inner Source implementation may provide many insights, we find that they lack a tool to assess the organization's level of Inner Source adoption. We see considerable scientific and practical use in working out characteristics of Inner Source

OpenSym '17, August 23–25, 2017, Galway, Ireland

ACM ISBN 978-1-4503-5187-4/17/08…$15.00
https://doi.org/10.1145/3125433.3125447

implementation based on insights from literature on Inner Source and OSS and assessing them in the software development department of a large medical diagnostics corporation. By creating a maturity model we define what an organization needs to implement when applying practices of Inner Source. The goal of this study is to develop a model which allows the evaluation of an organization's maturity level as regards their Inner Source implementation. Our model shows Inner Source practices derived from the literature of OSS as well as Inner Source. Applying our model, an organization can identify context-specific gaps regarding efforts made in implementing Inner Source and best practices derived from the literature and practical experience.

First we provide a literature review on what defines Inner Source in detail. Second, we describe our research method and our maturity model of Inner Source implementation. Third, we describe the results of the applied model within a large medical diagnostics company. Finally, we elaborate on the findings in the discussion section.

DEFINING INNER SOURCE

Within the scope of this study, we will refer to Inner Source as *"the leveraging of Open Source Software development practices within the confines of a corporate environment"* [28]. Unlike OSS projects which intend to make their source code available on the Internet, organizations wish to protect their intellectual property rights for competition and patent reasons [2].

However, a more detailed examination is needed of the aspect of adopting OSS development practices as described in the definition. There is no defined list of what these Inner Source practices are. There are, however, a number of common practices, which can be found within the literature: Robbins [23] promotes universal access to all project artifacts, opening the source code to all project participants and emphasizing the re-use of software components. Gurbani et al. [8] highlight the frequent releases and the management of contributions, Melian and Mähring the transparent development process [15] and Dinkelacker et al. the peer-review process [3].

Inner Source is best captured as a philosophy for decentralized software development, which is based on practices from OSS communities [29]. Adapting not only the practices, but also the tools from OSS development appears to be a straightforward undertaking. Furthermore, universal access to all development artifacts, such as the IDE and tools, is a precondition for developers within different business units being able to contribute to Inner Source. As regards OSS development tools, Robbins [24] in his study outlines how the use of OSS tools may also lead to the adoption of OSS methods. This underlines the importance of adequate tool usage when implementing Inner Source. Their main purpose is to complement, support or facilitate the Inner Source development effort [31]. Common examples of tools are: code repositories, wikis and issue tracking systems [23, 28], concurrent version control systems [9, 23, 25], a component reuse platform [16], a collaborative development program [14] or a software forge [21].

Similar to instances in which an OSS community collaboratively develops software, an Inner Source project must get enough people involved to create a favorable corporate culture [29]. This can be achieved by recognizing and rewarding collaborative behavior [3]. Universal access facilitates awareness throughout the entire team and more detailed knowledge of what people are working on and may help to improve coordination [10]. A corporate community will then develop what we call an 'OSS mentality'. This will lead to open discussion and transparency [6, 14] and voluntary contributions to project assets [28].

RESEARCH METHOD

The following section will show how our model was developed.

Initial Model

We reused some elements of the Capability Maturity Model of Integration for Development v1.3 (CMMI-DEV). The CMMI-DEV is a process improvement training and appraisal program providing guidance on applying best practices in a software development organization. The 22 process areas of the CMMI-DEV address the entire development process of an organization. As we have no intention of evaluating an organization's overall software development process, but rather their specific efforts related to Inner Source, we did not select any of the 22 process areas defined in the CMMI-DEV. Instead, our model is rooted to the three critical dimensions found by the Software Engineering Institute to be those that organizations typically focus on: People, Procedures & Methods and Tools & Equipment. These three dimensions are interrelated [1]. As an example, processes influence the tools and people involved in an organization and vice versa [1]. Processes, as stated, allow organizations to align the way they do business, they hold everything together. A focus on process provides the infrastructure necessary to maximize the productivity of people and the use of technology to be competitive [1].

In the CMMI-DEV, every process area has specific goals, as well as generic goals. The goals are then specified in greater detail with specific, generic and sub-practices and example work outcomes that describe the unique characteristics necessary to satisfy a process area [1]. This fits the extent of the CMMI-DEV, but not the scope of this study. In our model, the three dimensions all comprise a set of questions which address a practice that should be in place in order to satisfy a capability level.

A distinction is made in the CMMI-DEV between continuous and staged representation. Staged representation is concerned with selecting multiple process areas to improve; whether or not individual processes are performed is not the primary focus [1]. For the scope of our model, continuous representation seemed to fit best as it enables the

organization to choose the focus of its process improvement efforts by choosing those process areas that best benefit the organization.

Data Retrieval

A case study allows us to validate our model. Moreover, by applying our model to a concrete example, the reader of this case study will gain an insight into every construct upon which our model is based [4]. To this end, the use of qualitative methods in the form of semi-structured interviews will allow us to obtain the necessary insights. The seven interviews conducted are based on the questions of our model and are transcribed verbatim and analyzed using MAXQDA. The interviews were between 25 and 37 minutes long. Participants were distributed along the various occupation roles, ranging from Software Engineers to senior management. The participants' employment ranges between three and thirteen years, but the majority had prior work experience. We used the 32 questions of our main model to analyze the interviews. Therefore, the model's dimensions composed the categories and served as an initial point for the data analysis. This translated into twelve main categories (four capability levels for every of the three process areas) and a total of 32 categories (one for each question within the model). Further, we analyzed organizational documents concerning the Inner Source project, as well as policies governing OSS usage and legal aspects of the license review within the organization.

The selection of a case is an important aspect in case study research because the population defines the set of entities upon which the case study is built [36]. We chose the organization for two reasons: First, the organization started to implement Inner Source but is not finished yet. Thus, it provides an opportunity to analyze the organization throughout its ongoing process of implementing Inner Source. Second, the organization develops software globally. Developers do not necessarily know one another, but work for the same organization. Therefore, we see a fit to the theory of Inner Source which often takes place in a globally-distributed setting where knowledge sharing is seen as an important factor to speed up processes and innovation [14, 17].

Iteration Process

Our initial model was developed by analyzing literature on Inner Source; the results of the literature review served as an interview guideline [30]. Based on this guideline, we checked whether the model covers all the aspects necessary to evaluate the organization's Inner Source endeavor. The questions left room for the interviewees to talk about their own personal experiences. In the interviews and document analysis we found several important aspects around the Inner Source development practice that were not covered by our initial model. Furthermore, the data analysis revealed that some changes in the hierarchy of the questions led to better and more specific results for the evaluation. These additional aspects, as well as some changes in the hierarchy, were incorporated into a modified version of our model. We would apply the same procedure in future evaluations to allow us to continuously improve the model described in our paper. As an example, in our literature-based initial model, Q26 (firm policy on tools) did not exist. After analyzing the interviews and documents, we found that such a policy is important because of legal compliance. It ensures that the risks of OSS usage within products are properly identified, addressed and mitigated.

Evaluation and Classification

In order to evaluate and classify an organization's efforts to implement Inner Source, we develop an evaluation-method. For every question, points ranging from zero to four can be awarded. These points allow us to determine whether or not a capability level is satisfied. An average of 75% is needed to satisfy a level. Each question tries to find evidence for the presence of a certain practice, know-how or tool. The first *two points* are awarded if a question can be confirmed i.e. if know-how, a practice or tool is present. A *third point* is awarded if there is plenty of evidence for said practice, know-how or tool, that is, there are multiple individuals or examples confirming not only its presence but also its sophisticated use. The *fourth point* is awarded when know-how, a practice, or tool is performed or used to a level which is comparable to the "best practice" found in the literature and there is little to no room for further improvement. Two researchers made the classification separately and discussed the results.

Brief Description of the Case

The analyzed organization is a world-leading provider of medical diagnostic system solutions. It develops analysis systems for laboratory diagnostics and provides global services and support. The organization develops the hardware used in the diagnostic instruments, as well as the software that controls and operates them. The Inner Source endeavor within the organization is built mainly around an integrated software platform of reusable components. It is therefore the core asset base for developing diagnostics instruments and laboratory information technology solutions. The goal of the platform is to accelerate product development with high quality and code re-use. By hosting many different hardware and software components, the platform serves as a large scale re-use-platform and aims to reduce the workload of new software development projects. The platform provides more than 10 different hardware and software elements for standalone use and serves as a basis for future products and projects. As it provides many different software development kits, such as libraries, architecture patterns, documentations, codes and wikis, the platform serves as a collaborative development platform for many different development teams. Currently, around 70 globally distributed software developers, software architects and managers are working on the platform.

RESULTS

Each critical dimension and the corresponding capability levels will now be described in detail.

People Dimension

This dimension focusses on OSS development practices within an organization. There are practices that take place either on an individual level or within a community. The goal of this dimension is to find out how individuals perceive and live the OSS phenomena within a firm, how they interact with coworkers and if there is a prospering community to be found.

A0: Incomplete - This capability level determines the basic knowledge and understanding of OSS [19]. As implementing Inner Source entails adopting OSS practices, an organization's developers first need a basic understanding of OSS to successfully adopt Inner Source.

A1: Performed - Inner Source cannot be enforced, it needs voluntary participation [36] and employees who engage in the exchange of ideas and information [25, 35]. Such a culture of exchange cannot be created solely by the employees and developers. The leadership must be involved [21] to facilitate and foster open discussion [14].

A2: Managed - A practice of OSS development is its liberal task selection. In an OSS setting, developers can contribute to any given development artifact due to the universal access to the source code [13, 21]. Developers should be able to contribute or follow other projects than the ones they are assigned to [9, 14, 28]. The management should support the company's Inner Source endeavor and support the OSS mentality [3, 8, 21]. Increased management support should induce adequate guidelines and regulations and therefore culminate in a clear Inner Source strategy [2, 3, 29].

A3: Defined - OSS development is driven by various motives such as the desire to learn new skills, to create new features deemed necessary or to enjoy the freedom to contribute to other projects [33]. To fully utilize the potential of Inner Source, the goal is to involve as many developers as possible [21]. Therefore, an Inner Source culture must be facilitated. To do so, the organization needs long-term goals and a vision that includes Inner Source [29].

Procedures & Methods Dimension

This dimension integrates procedures and methods which are oriented around software development, such as requirements engineering. Procedures and methods that take place through and within the community - such as collaborative development and peer-review - are also necessary.

B0: Incomplete - Although requirements elicitation might differ between OSS development and Inner Source it is fundamental to an organization how requirements are gathered. This capability level aims to find out how the organization gathers requirements. Further, the organization should provide basic information about OSS to their employees [9].

B1: Performed - The possibility to review and contribute to all project artefacts is a fundamental element in OSS projects [13, 21]. This capability level evaluates whether developers have read-access to software projects or components within their organization [14]. Further, developers should also be able to contribute to projects by sending bug reports and feature requests [7, 11]. Moreover, an organization should devote effort into gathering and analyzing (non-functional) requirements [9, 28]

B2: Managed - Making software components re-usable can speed up the development process and reduce cost [22]. It can also increase the number of shared assets within an organization [28]. Further, this capability level investigates the extent to which developers have commit access that allows them to contribute source code [9, 28].

B3: Defined - There should be a process which ensures the review of proposed features and improvements [27]. Peer-review is seen as an important OSS practice [8, 15, 27], because it may provide many useful inputs and uncover flaws and becomes increasingly effective as more developers are involved [19]. An organization should also make use of its established components and pursue large-scale code re-use [22].

Tools & Equipment Dimension

The third dimension focusses on the technological aspects of Inner Source and covers the OSS tools and components. The progression through the levels is based mainly around tools that support collaboration and development. Because Inner Source is not a completely new software development approach and is somewhat related to agile software development [34], the tools are not unique to Inner Source.

C0: Incomplete - Since developers need to use various tools in order to adopt OSS development practices, an organization must allow its use. Because OSS and Inner Source development are often geographically distributed [12], this capability level determines the basic conditions for the use of OSS tools within the organization.

C1: Performed - OSS projects often use code repositories, wikis, mailing lists and an issue tracking system [7, 23, 28]. Such tools facilitate communication, knowledge-sharing and provide feedback [28] and are also involved when developing in an Inner Source setting.

C2: Managed - In order to make contribution easier, a common set of development tools should be defined [27]. A firm policy on what tools or components can be used help to reduce the number of tools and components [26]. A tool commonly-used in OSS project is a version control system [23, 27] or change management system [8]. Further, a component-reuse-platform can help to manage and reuse components [23].

C3: Defined - A collaborative development environment can help to enable developers to collaborate organizational-wide [3] and reduce the effort of starting new projects by providing complete and standard toolsets [23], often called Software Forges [21]. To leverage the potential of contributors, an organization should consider expanding the

Inner Source term such as to engage in *Controlled Source* [3] and share software code with third parties under a non-disclosure agreement [14]. A step further would be to reveal source code to the general public [3]. This could be due to various reasons such as to profit from OSS communities by sharing development and maintenance cost [14, 31].

Table 1 shows our model. It is a 4-by-3 matrix where the columns represent the three dimensions: People, Procedures & Methods and Tools & Equipment. Each dimension has four capability levels represented by the rows. The model is based on hierarchy, simpler practices are located at lower capability levels and more complex practices are at higher levels. The questions within every element of the matrix represent a particular practice found in the OSS or Inner Source literature.

	A: People Governance, community, individual	**B: Procedures & Methods** Requirements engineering, development method	**C: Tools & Equipment** Development tools, open source components
0: Incomplete	Q1: Is there know-how available about open source software within the company? [19] Q2: Is there know-how available about the open source mentality within the company? [19]	Q12: Is there information about OSS within the Organization? [9] Q13: Is there an open discussion on requirements elicitation? [26]	Q23: Are developers allowed to use OSS development tools? [3, 29] Q24: Is there a tool that allows instant messaging between developers? [12]
1: Performed	Q3: Is there a mentality of open discussion for projects? [14] Q4: Is there a frequent exchange of ideas/problems? [9, 25, 35] Q5: Is there a general interest for the Open Source phenomena from co-workers who are involved in software development? [19]	Q14: Do developers have read-only access to the source code of any software product or component within the company? [14] Q15: Can anyone contribute bug reports/feature requests to software products? [7] Q16: Is there effort devoted to gathering and analyzing (non-functional) requirements? [9, 28]	Q25: Are there supporting tools for collaborative software development like: code repositories, wikis, mailing lists or issue trackers? [7, 23, 28] Q26: Is there a common set of development tools or firm policy on what tools should and/or could be used? [26, 27]
2: Managed	Q6: Are there developers that contribute to other products/assets than the ones they are assigned to? [9, 14, 28] Q7: Are there enough regulations/guidelines to say that the company has an Inner Source strategy? [2, 3, 29] Q8: Does the management level support the Inner Source endeavor? [3, 9, 29]	Q17: Is there a process for developers receiving commit access to contribute to software products? [28] Q18: Can a developer contribute a feature or improvement because the developer perceives it as useful/helpful? [9] Q19: Is there a process to make software components reusable within the company? [22]	Q27: Is there some sort of a version control system or a change management system? [9, 23, 27] Q28: Is there a platform to manage reusable components? [23] Q29: Is there a (top down) endeavor towards the use of OSS? [27]
3: Defined	Q9: Is there a community notion that drives developers to contribute to the firms' general welfare and/or projects/assets other than the ones they are assigned to? [9] Q10: Are there enough people involved in and around inner/open source activities and ideas for a corporate culture around that phenomenon? [25, 29] Q11: Does the company have an inner/open source vision? [29]	Q20: Is there a process to ensure a fast review and identifying of proposed requirements and improvements? [8, 15, 27] Q21: Is there a predefined process to ensure quick turnaround of peer-reviews to resolve problems quickly? [8, 27] Q22: Is there ambition towards or a process of large-scale code/ software/ module-reuse within the company? [22]	Q30: Is there a collaborative development program / software forge? [14, 21, 23] Q31: Does the company reveal source code of software products or components to third parties under a non-disclosure agreement? [3, 9, 14] Q32: Does the company reveal source code of software products or components with an OSS license? [3, 31]

Table 1. Maturity Model of Inner Source Implementation.

DISCUSSION OF THE FINDINGS

Based on our evaluation, the organization has reached the *performed* level in the People dimension and the *managed* level in the Procedures & Methods and Tools & Equipment dimension. Fig. 1 illustrates the results of testing our maturity model.

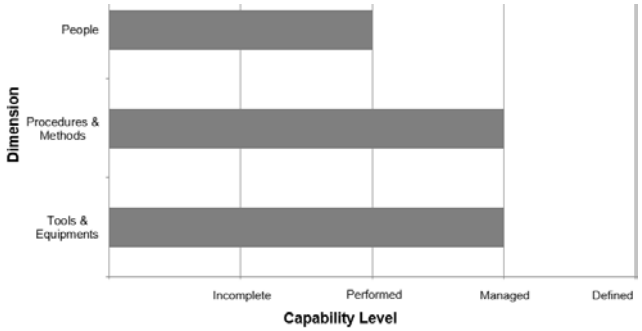

Figure 1. Reached capability levels of the organization.

By applying our model to an organization, we identified various gaps between best practices found in the literature and the efforts made by the organization. The organization did not satisfy the managed and defined level in the People dimension as well as the defined level in the Procedures & Methods and Tools & Equipment dimension. Based on our model, we have found the following gaps:

People: There is a lack of know-how exchange between projects to which a developer has not been assigned. It is possible to contribute to other products than a developer is assigned to, but it is not typical. The kind of assistance does not usually result in usable code. Developers are often assigned to a specific project and tend to stick to it. There are exchanges of know-how, as one interviewee stated: *"Besides the management, we ourselves saw the benefit of sharing knowledge. That's what we are trying to promote, not only with ourselves, but also with our colleagues."* However, it is not possible to provide assistance on a long-term basis to a project to which a developer is not assigned. Because there are not many examples of contributions to other projects, there is still much room for improvement in that regard. The organization needs to improve collaboration by getting developers comfortable with sharing code and providing assistance across the organization. Regarding the presence of a clear Inner Source or OSS strategy, there is a complete lack. There are some guidelines and rules on the use of OSS, but a top-down strategy is missing.

Procedures & Methods: Although the platform comprises various projects, it is still an active decision for a project owner to develop a project with the platform (and therefore make it re-usable) or not. A development project leader formulated it as follows: *"If we want to make it re-usable then there is an active decision to do so. […] The new components must present potential value for future projects and needs. Then, there must be free resources within the platform to implement it."* To exploit the full potential of a re-use platform, as many projects and developers as possible should be embedded into the platform, but there are no concrete plans in that regard as of yet. To this end, the organization could decide to set the platform as mandatory for every new project to enlarge the pool of possible contributors.

Tools & Equipment: There are some examples in which code and components were shared with third parties. This was mostly done when working with an external company or a subsidiary. Such collaborations were always regulated under a non-disclosure agreement. A software developer described such collaboration: *"We do collaborate with external companies, where we share repositories or source code and where they can contribute to our code."* However, regarding the revealing of source code under an OSS license, no examples or any intention to do so were found. The organization could profit by revealing source code under an OSS license in various ways. A strategy that controls when revealing source code is allowed or desired is missing. The organization would first need to develop such a strategy.

CONCLUSION AND FURTHER RESEARCH

This paper contributes to current research by providing a model designed to evaluate an organization's efforts in implementing Inner Source. The model is derived from both OSS and Inner Source literature and provides an overview of the literature concerning the dimensions of People, Procedures & Methods and Tools & Equipment. These three dimensions are targeted by our model with a total of 32 questions evaluating the efforts by an organization in implementing Inner Source.

Based on our results, the organization is now able to work on specific gaps to improve the implementation of Inner Source. By introducing Inner Source, the organization was able to achieve a faster time-to-market as well as to make software components re-usable. As stated in the interviews, if a new device is similar to an existing one, up to 30% of the source code can be re-used. Through the transparent and open development process introduced with Inner Source, collaboration between different project teams and employees across geographically distributed locations improved. Moreover, Inner Source enabled bottom-up initiatives from individuals, resulting in regular workshops around GIT and Linux. With the introduction of Inner Source, requirements are discussed in more detail with all project leaders in weekly meetings. To sum up, the organization was able to improve their software development process using Inner Source.

However, the implementation of Inner Source can be a quite challenging process. Through our model and the gaps identified when applying our model, the analyzed organization is now able to further improve their Inner Source implementation by targeting their efforts to bridging the gaps, thereby gaining greater benefits. With our model, organizations have the potential to overcome many of the challenges posed by Inner Source by evaluating its adaption comprehensively and revealing current shortcomings. An evaluation using our model presents organizations with the

necessary insights to specifically address shortcomings within their Inner Source implementation and therefore efficiently improve it in order to fully profit from Inner Source and all the advantages it brings.

However, there are some limitations to our research. First, our model was applied only to one case based on data of seven conducted interviews as well as organizational documents. However, the data was substantial with almost 200 pages of transcript and represents a sophisticated example of an Inner Source implementation in a world-leading organization. Further applications of our model could help to overcome this limitation. We expect the model to be applicable to other cases. Applying it to other case studies could reveal that some changes in the hierarchy of the questions or new questions might lead to better and more specific results. We would apply the same procedure in such evaluations so that the model described in our paper can be improved continuously. Therefore, we would welcome other researchers applying our model.

Second, although we carried out a classification, there is still some room for interpretation when awarding points to a question. To reduce this limitation, two researchers made the classification separately and discussed the results. Third, the nature of the maturity model itself brings a limitation. If one were to strictly follow the CMMI-DEV, each question within each capability level should have clearly defined goals (specific & generic goals) that need to be satisfied in order to reach the next level. Each goal is again rated on a 6-level implementation scale, providing a score that determines whether a goal is satisfied. Because the CMMI-DEV model is a basic reference only, we altered and reduced the model to fit the purpose and scope of this study.

We hope that further research will be able to shed light on the motivation of developers' engagement in Inner Source implementation. If an organization is able to improve motivation to contribute to Inner Source, the organization as a whole could benefit.

REFERENCES

[1] Carnegie Mellon University, Software Engineering Institute 2010. *CMMI for Development v1.3*.

[2] Dinkelacker, J. and Garg, P.K. 2001. Corporate Source: Applying Open Source Concepts to a Corporate Environment (Position Paper). (Toronto, Canada, 2001).

[3] Dinkelacker, J., Garg, P.K., Miller, R. and Nelson, D. 2002. Progressive open source. *Proceedings of the 24th International Conference on Software Engineering* (2002), 177–184.

[4] Eisenhardt, K.M. 1989. Building Theories from Case Study Research. *The Academy of Management Review*. 14, 4 (Oct. 1989), 532–550.

[5] Fitzgerald, B. 2006. The transformation of open source software. *MIS Quarterly*. (2006), 587–598.

[6] Gaughan, G., Fitzgerald, B. and Shaikh, M. 2009. An Examination of the Use of Open Source Software Processes as a Global Software Development Solution for Commercial Software Engineering. (2009), 20–27.

[7] Grammel, L., Schackmann, H., Schröter, A., Treude, C. and Storey, M.-A. 2010. Attracting the Community's Many Eyes: An Exploration of User Involvement in Issue Tracking. *Human Aspects of Software Engineering* (New York, USA, 2010), 3:1–3:6.

[8] Gurbani, V.K., Garvert, A. and Herbsleb, J.D. 2006. A case study of a corporate open source development model. *Proceedings of the 28th international conference on Software engineering* (2006), 472–481.

[9] Gurbani, V.K., Garvert, A. and Herbsleb, J.D. 2005. A case study of open source tools and practices in a commercial setting. *ACM SIGSOFT Software Engineering Notes*. 30, 4 (2005), 1–6.

[10] Gutwin, C., Penner, R. and Schneider, K. 2004. Group awareness in distributed software development. *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (2004), 72–81.

[11] Heppler, L., Eckert, R. and Stuermer, M. 2016. Who cares about my feature request? In: *IFIP International Conference on Open Source Systems*. Springer, Cham, 2016. 85-96.

[12] Herbsleb, J.D. and Mockus, A. 2003. An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on*. 29, 6 (2003), 481–494.

[13] Lindman, J., Rossi, M. and Marttiin, P. 2008. Applying open source development practices inside a company. *Open Source Development, Communities and Quality*. Springer. 381–387.

[14] Melian, C., Ammirati, C.B., Garg, P. and Sevon, G. 2002. *Building Networks of Software Communities in a Large Corporation*. Citeseer.

[15] Melian, C. and Mähring, M. 2008. Lost and gained in translation: Adoption of open source software development at Hewlett-Packard. *Open Source Development, Communities and Quality*. Springer. 93–104.

[16] Mockus, A. 2007. Large-scale code reuse in open source software. *Emerging Trends in ICSE'07 Intl. Workshop on Emerging Trends in FLOSS Research and Development*, Minneapolis, USA, 2007.

[17] Morgan, L., Feller, J. and Finnegan, P. 2011. Exploring inner source as a form of intra-organisational open innovation. (2011), 1-12.

[18] Neus, A. and Scherf, P. 2005. Opening minds: Cultural change with the introduction of open-source collaboration methods. *IBM Systems Journal*. 44, 2 (2005), 215–225.

[19] Raymond, E.S. 2001. *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary*. O'Reilly Media, Inc.

[20] Riehle, D., Capraro, M., Kips, D. and Horn, L. 2015. *Inner Source in Platform-Based Product Engineering*.

tech. report CS-2015-02, Dept. of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany.

[21] Riehle, D., Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi, Y., Naveh, B. and Odenwald, T. 2009. Open collaboration within corporations using software forges. *Software, IEEE*. 26, 2 (2009), 52–58.

[22] Riehle, D. and Kips, D. 2012. *Geplanter Inner Source: Ein Weg zur Profit-Center-übergreifenden Wiederverwendung*.

[23] Robbins, J. 2005. Adopting open source software engineering (OSSE) practices by adopting OSSE tools. *Perspectives on free and open source software*. (2005), 245–264.

[24] Robbins, J.E. 2002. Adopting OSS methods by adopting OSS tools. *CollabNet, Inc*. (2002).

[25] Sharma, S., Sugumaran, V. and Rajagopalan, B. 2002. A framework for creating hybrid-open source software communities. *Information Systems Journal*. 12, 1 (2002), 7–25.

[26] Stol, K.-J. 2011. Supporting product development with software from the bazaar. (2011).

[27] Stol, K.-J., Avgeriou, P., Babar, M.A., Lucas, Y. and Fitzgerald, B. 2014. Key factors for adopting inner source. *ACM Transactions on Software Engineering and Methodology*. 23, 2 (Apr. 2014), 1–35.

[28] Stol, K.-J., Babar, M.A., Avgeriou, P. and Fitzgerald, B. 2011. A comparative study of challenges in integrating Open Source Software and Inner Source Software. *Information and Software Technology*. 53, 12 (Dec. 2011), 1319–1336.

[29] Stol, K.-J. and Fitzgerald, B. 2015. Inner Source—Adopting Open Source Development Practices in Organizations A Tutorial. (2015).

[30] Turner III, D.W. 2010. Qualitative interview design: A practical guide for novice investigators. *The qualitative report*. 15, 3 (2010), 754.

[31] Van Der Linden, F. 2009. Applying open source software principles in product lines. *Upgrade*. 10, (2009), 32–41.

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

[33] Von Krogh, G., Haefliger, S., Spaeth, S. and Wallin, M.W. 2012. Carrots and rainbows: Motivation and social practice in open source software development. *MIS quarterly*. 36, 2 (2012), 649–676.

[34] Warsta, J. and Abrahamsson, P. 2003. Is open source software development essentially an agile method. *Proceedings of the 3rd Workshop on Open Source Software Engineering* (Portland, Oregon, 2003), 143–147.

[35] Wenger, E.C. and Snyder, W.M. 2000. Communities of practice: The organizational frontier. *Harvard Business Review*. 78, 1 (2000), 139–146.

[36] Wesselius, J. 2008. The bazaar inside the cathedral: Business models for internal markets. *Software, IEEE*. 25, 3 (2008), 60–66.