Exploring Open Source Software as an Enabler for Open Innovation in Software-Intensive Organizations

Hussan Munir

Licentiate Thesis, 2015
Department of Computer Science
Lund University

Licentiate Thesis 3, 2015
ISSN: 1652-4691

Department of Computer Science
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: hussan.munir@cs.lth.se
WWW: http://cs.lth.se/hussan_munir

Printed in Sweden by Tryckeriet i E-huset, Lund, 2015

ABSTRACT

Background: *Open Innovation (OI)* has attracted scholarly interest from a wide range of disciplines since its introduction by Henry Chesbrough in 2003. However, OI remains unexplored for software-intensive organizations and its potential impact on the organization's innovative performance. There are many ways of enabling OI in software-intensive organizations and *Open Source Software (OSS)* is a one example.
Aim: The aim is to investigate how OSS projects enables innovation for organization's proprietary solutions. Specifically, OI needs to be explored at the project level instead of the firm's level to identify its potential impact on the firm's innovative performance. Consequently, we can eradicate other confounding factors that may or may not contribute to firms innovative performance thereby, making it difficult for the researchers to understand the impact of OI on firms innovative performance.
Research Methodology: The thesis followed an empirical approach and the opted research methodology encompasses a systematic mapping study and a survey followed by two case studies. We investigated OI by studying OSS projects namely Jenkins and Gerrit.
Results: Start-ups have higher tendency to adopt OI compared to market leading firms and too open behavior from incumbents may have negative impact on the firm's innovative performance. Moreover, the case company [Sony Mobile] set up a Tools department to work with communities, which lead to more inner source initiatives in other units of Sony. However, the project openness remains limited to the non-competitive tools which is not the direct source of revenue. The whole transition from *Closed Innovation* to *Open Innovation* model was a paradigm shift to get close to Google's tools chain, driven bottom up by engineers with the support of management. Moreover, OI testing process does not strictly adhere to standard Software Engineering testing process. Both requirements and issues are prioritized based on the most pressing needs of an organization.
Conclusion: As a result of adopting OI, the case company freed-up developers time, better quality assurance, faster releases and upgrades. For adopters of OI, it should be used as a complementary approach to speed up the firms internal innovation processes rather than replacing them.

ACKNOWLEDGEMENTS

This work was funded by Synergies project, grant 621-2012-5354 from the Swedish Research Council.

First and foremost, I want to thank the Almighty Allah (God) because he has graced my life with opportunities that I know are not of my hand or of any other human hand. He has always shown me the way through crests and troughs of life and he is the one I always look up to.

I submit my highest appreciation to my supervisors Prof. Dr. Per Runeson and Dr. Krzysztof Wnuk for their patience, immense knowledge and continuous support. I could not have imagined having better supervisors and mentors for my ongoing PhD. study. I am also indebted to Dr. Kai Petersen for kick starting my research career prior to starting my PhD. Sony Mobile Lund also deserves a lot of credit for sharing empirical data to make my research industry relevant.

I am also thankful to all of the Department of Computer Science faculty members and the Software Engineering Research Group for their help and support. In particular, I would like to thank Johan Linåker for our ongoing research collaboration and amusing conversations both on and off the work. Furthermore, it is also impossible to have a good week at work without having a Sushi with Alma and Mehmet.

All my academic growth would not have been possible without the support of my younger brother and an elder sister. She played a special role in my early educational career when I needed the encouragement and self-confidence. Last but not least, I am eternally grateful to my parents for the disciplined childhood, sacrifices, unceasing encouragement, selfless support and attention throughout my life. I and you (Mom and Dad) know that your kid would not have come that far in his life without you.

Hussan Munir

LIST OF PUBLICATIONS

In the introduction chapter of this thesis, the included and related publications listed below are referred to by Roman numerals.

Publications included in the thesis

I **Open Innovation in Software Engineering: A Systematic Mapping Study**
Hussan Munir, Krzysztof Wnuk and Per Runeson
Empirical Software Engineering, Pages 1-40, 2015.

II **A Survey on the Perception of Innovation in a Large Product-focused Software Organization**
Johan Linåker, Hussan Munir, Per Runeson, Björn Regnell, Claes Schrewelius
6th International Conference on Software Business-ICSOB, 2015.

III **Open Innovation through the Lens of Open Source Tools: An exploratory case study at Sony Mobile**
Hussan Munir, Johan Linåker, Krzystof Wnuk, Per Runeson and Björn Regnell
Submitted to a Software Engineering Journal, 2015 (Under review).

IV **Software Testing in Open Innovation: An exploratory case study of Acceptance Test Harness for Jenkins**
Hussan Munir, Per Runeson
Proceedings of the 2015 International Conference on Software and System Process (ICSSP).

Related Publications

V **Considering Rigor and Relevance When Evaluating Test Driven Development: A Systematic Review**
Hussan Munir, Misagh Moayyed, Kai Petersen
Information and Software Technology., vol. 56, no. 4, Pages 375-394, 2014.

VI **An Experimental Evaluation of Test Driven Development vs. Test-last Development with Industry Professionals**
Hussan Munir, Krzysztof Wnuk, Kai Petersen and Misagh Moayyed
Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering EASE 2014.

Contribution statement

All papers included in this thesis have been co-authored with other researchers. The authors' individual contributions to Papers I-IV are as follows:

Paper I
For Paper I Hussan Munir was the lead author. He designed and executed the study followed by a validation and paper review from Dr. Krzysztof Wnuk and Prof. Per Runeson. Hussan Munir was responsible for data collection and analysis. Moreover, Dr. Krzysztof Wnuk was involved in the screening of studies extracted from databases.

Paper II
Johan Linåker and Hussan Munir were responsible for writing the paper after performing the analysis of data. Dr. Krzysztof Wnuk and Prof. Per Runeson were involved in the initial planning and distribution of survey. The results were than reviewed by the all four authors.

Paper III
Hussan Munir is the first author with the main responsibility for the research effort together with Johan Linåker. Hussan Munir and Johan Linåker wrote a majority of the text after performing the data mining and data analysis, and the co-authors contributed with constructive reviews. Dr. Krzystof Wnuk was also involved in conducting the interviews with industry professionals.

Paper IV
Paper IV was performed by Hussan Munir with Prof. Per Runeson. Hussan Munir was responsible for the study design, data collection, analysis of data, and did most of the writing. Prof. Per Runeson provided feedback during the study and reviewed the paper.

CONTENTS

Introduction **1**
1 Introduction 1
2 Related work and terminology 3
3 Research goals 4
4 Research methodology 6
5 Results and synthesis 8
6 Ethical aspects and threats to validity 15
7 Future work 16

Included papers **19**

I Open Innovation in Software Engineering: A Systematic Mapping Study **21**
1 Introduction 22
2 Related work 23
3 Research methodology 28
4 Results and analysis 34
5 Discussion 55
6 Implications for Research and Practice 58
7 Conclusions 60

Appendix A Rigor and Relevance Criteria **61**
1 Rigor 61
2 Relevance 62

Appendix B Database search strings **65**

II A Survey on the Perception of Innovation in a Large Product-focused Software Organization **67**
1 Introduction 67
2 Related work 69

3 Methodology 70
4 Results 73
5 Conclusions 81

III Open Innovation through the Lens of Open Source Tools: An exploratory case study at Sony Mobile 83
1 Introduction 84
2 Related work 86
3 Case study design 88
4 Quantitative analysis 97
5 Qualitative analysis 100
6 Discussion 109
7 Conclusion 117

IV Software Testing in Open Innovation: An Exploratory Case Study of the Acceptance Test Harness for Jenkins 119
1 Introduction 119
2 Research Design 120
3 Results and Analysis 122
4 Conclusions 127

Bibliography 129
References 132

INTRODUCTION

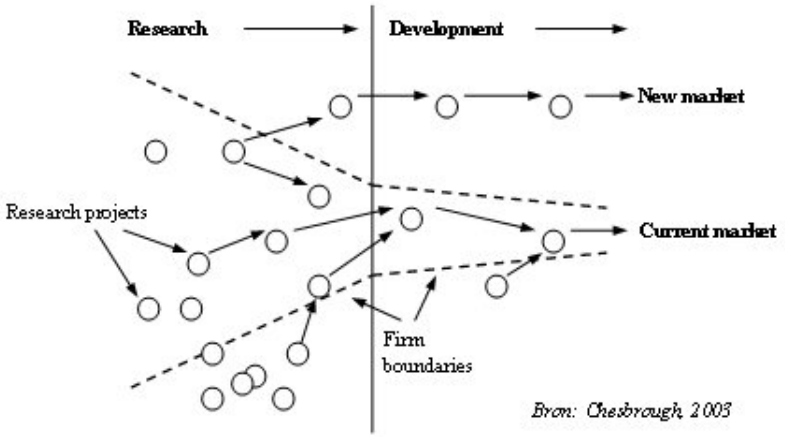
1 Introduction

Open innovation (OI) is an emerging management paradigm which originated from high technology industry practices in the US and Japan [23]. OI can be traced back to Allen's [7] collective invention in the 19th century. Two decades after Allen's paper from 1983, Henry Chesbrough coined the term *Open Innovation* as *"a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology"*. In Fig.1, the dotted line in the funnel shows the boundary of the firm where ideas can seep in and out. Ideas can originate from inside the firm's research process, but some of ideas may seep out of the firms, either in the research stage or later in the development stage. One of the typical examples of idea leakage is a start-up company, often initiated by some of the company's own personnel. Ideas can also start outside the firm's own labs and can move inside.

As the world moves towards the *Knowledge Society* [38], utilizing knowledge and ideas from both inside and outside of an organization becomes more important than ever. Therefore, OI can be analyzed and researched from the various management perspectives: strategic positioning, business models, value chain, knowledge creation and core competencies. OI initiated an unabated interest among researchers in innovation management [68], economics, psychology, sociology, and also Software Engineering [136]. The work initiated by Chesbrough inspired both scholars and practitioners to rethink the design of the innovation strategies in a networked environment [68]. OI encompasses various forms of knowledge transfer such as inbound (outside-in knowledge), outbound (inside-out knowledge), and coupled process (outside-in and inside-out knowledge) [50].

The novelty of OI was questioned by an argument that closed innovation might have been the exception in the history, characterized mostly by open innovation practices [104]. In response, Chesbrough proposed *Erosion Factors* that undercuts the logic of the *Closed Innovation* model of R&D and developed the logic of the *Open Innovation* model. These erosion factors, such as increased mobility

Figure 1: Open Innovation



of workers, declining US hegemony, more capable universities, growing access of start-up firms to venture capital, changed the conditions under which firms innovate. In addition, the rise of internet has brought the knowledge access and sharing capabilities of previous firm-specific internal network to the World Wide Web [23].

In the field of software engineering, the success of *Open Source Software (OSS)* in the past twenty years indicates its existence before the term OI was coined [91]. Furthermore, OSS is the most straight forward application of OI software development and used as an example of OI in the studies included in this thesis [22]. Both OSS and OI tend to favor the use of external knowledge together with internal knowledge as a mutually beneficial measure for organizations and communities. However, it should be noted that OSS is *not* equivalent to OI. The underlying difference between OI and OSS is that the the open innovation efforts are usually run by one company trying to innovate by reaching the knowledge outside its walls. The problem to be solved or the opportunity ((e.g. Build Failure Analyzer, Gerit Jenkins Trigger)) to be addressed is *owned* by the company running the open innovation project and with the company at the center of all collaboration. The solution to the problem may have a direct on indirect impact on firm's innovation process. On the contrary, OSS problem or opportunity itself is the central point of focus, so people and organizations all connect to each other rather than working through one central organization. Furthermore, OI company has the business model influenced or driven by the OI definition. The business model utilizes both internal and external ideas to create value, while defining internal mechanism to claim some portion of the value.

The OI vs. OSS phenomenon can be explained by Linux development when IBM donated hundreds of patents and invested more than $100 million a year

to support the Linux OS. One of the OI advantages is that the risks and costs of development can be shared among the stakeholders. Although, IBM invested significant amount of money in the Linux development, other firms such as Nokia, Hitachi and Intel also made substantial investments as well [90]. By supporting the Linux, IBM was strengthening its own business model in selling proprietary solutions for its clients running on top of Linux. Additionally, the openness of Linux also gave IBM more freedom to co-develop products with its customer [23].

However, the shift from the *Closed innovation* to the *Open innovation* model poses significant challenges to software-intensive organizations in terms of keeping their competitive advantage in relation to their competitors. The openness challenges software intensive organizations on both operational and strategic levels. This thesis focuses on investigating the OSS projects considered representative examples of OI to investigate the impacts of OI on firms innovative performance. Particularly, what triggers software-intensive organizations to indulge themselves in the OI enabled projects and the innovation outcomes attached to it. Furthermore, it is desirable to see if the existing practices of requirements engineering and testing fits well with Open Innovation. The four studies in the thesis are exploratory in nature due to lack of existing evidence on OI in software engineering.

2 Related work and terminology

Despite the wide interest in several domains, OI is far from thoroughly researched in software engineering. OSS is often explored as one of the main examples of OI in order to incorporate the external knowledge and innovation to internal product innovation. However, Munir et al. [107] recognized the lack of systematic efforts to summarize and synthesize the state of the research on OI in software engineering. The previously attempted reviews were either partly systematic [68, 141, 150] or focused on the metrics used to measure innovation in OSS [40].

Organization use different strategies to engage in OSS communities [32], e.g. adopting selective revealing [62] or OI models [21]. West et al. [143] highlighted the strategies that organizations use to acquire, incorporate the external knowledge into their internal innovation processes and exploiting the *Intellectual Property Rights (IPR)* by a selective revealing strategy. Stuermer et al. [132] conducted a study on applying the private collective model at Nokia to identify the incentives for individuals investing in OSS and the firms. Nokia benefited from the introduced private collective model in terms of learning effects, reputation gain, reduced development effort and low knowledge protection costs. On the other hand, the cost of implementing the private collective model entails difficulty to differentiate, guard business secrets, reduce the community barriers and give up organizational control.

In addition, OI entails challenges on process and business levels. West et al. [145] highlighted the business related challenges faced by the leading firms

in the development of Symbian: 1) balancing the interests of all stakeholders, 2) knowing the requirements for a product that has yet to be created, and 3) prioritizing the conflicting needs of all stakeholders. On the other hand, Conboy and Morgan [27] hinted upon process related challenges that agile and OI do not get along well in terms of dealing with the management of innovative requirements and release planning [27].

Software intensive organizations (see Table 1) intended to indulge themselves in OSS communities, need to adjust their software development processes in their efforts to fix bugs and contribute new features to the community. These efforts might reduce the maintenance cost compared to in-house software development. Furthermore, OSS involvement may also entails different modes of working in terms of separate requirements engineering process for innovative requirements [87, 148] and OSS governance mechanism [90] to facilitate software development in OI context. Dahlander [34] concluded that initiating an OSS project is often a pragmatic way of attracting skilled workforce from communities. Moreover, having a dedicated employee working close to the community seems to be an enabler for not only building a good reputation of an organization in the community, but also allow exercising the governance/control mechanism to steer the development towards organization's business model. Linden et al. [96] concluded that when a software product loses its competitive value in terms of profitability, customers, innovation and learning [76] with the passage of time due to improvements and ever growing size of the software, it becomes a good candidate for OSS development.

Furthermore, it remains unclear what motivates software-intensive organizations and what are the key factors these organizations consider before adopting OI. Chesbrough [23] pointed out that convincing evidence with the large sample studies on the performance improvement from OI is still lacking. Therefore, there is a need to perform case studies on a product level instead of a firm's level in order to see OI impact on firm's internal innovation.

3 Research goals

To better understand OI in software engineering, the following *Research Goals (RG)* are formulated (see Fig. 2).

RG1: To synthesize the research knowledge on OI for software-intensive organizations

RG2: To understand the perception of innovation in large scale software-intensive organizations through an industrial survey

RG3: To explore and evaluate how software-intensive organizations use OSS as an enabler for OI and innovation outcomes

RG4: To investigate how software testing is performed in OI

Table 1: Definitions

Terms	**Definition**
Software-intensive organization	Organizations extensively using or developing softwares with the focus on adopting OSS that influence organization's innovation capacity.
Acceptance test harness	It is part of the Jenkins project used to test (unit tests) Jenkins plugins in an automated fashion [106].
Jenkins	Jenkins is the leading open source continuous integration server. It provides 1000+ plugins built in Java to support building and testing [2].
Gerrit	It is a web-based code review tool built on top of the git version control system [3].
Product innovation	Product innovation is the introduction of a good or service that is new or significantly improved with respect to its characteristics or intended uses [4].
Process innovation	Process innovation is the implementation of a new or significantly improved production or delivery method [4].
Business innovation	Business innovation is the implementation of a new marketing method involving significant changes in product design or packaging, product placement, product promotion or pricing [4].
Organizational innovation	Organizational innovation is the implementation of a new organizational method in the firm's business practices, workplace organization or external relations [4].

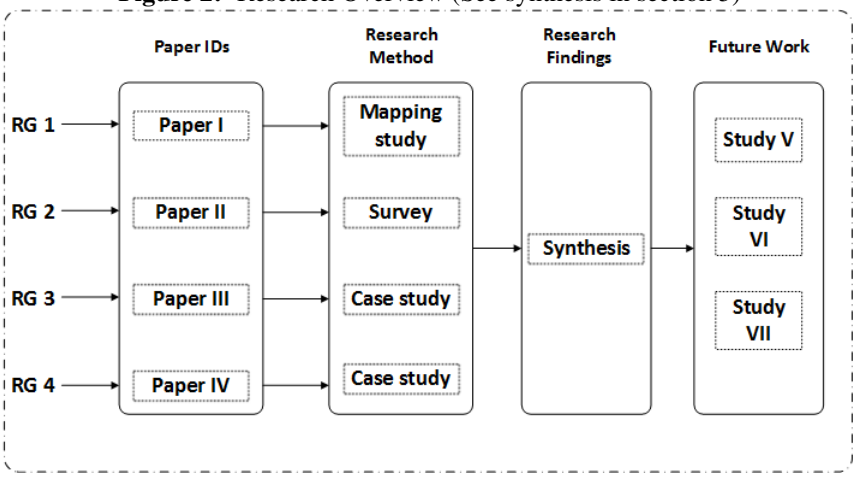
As can be seen in Figure 2, *RG1* triggers *Paper I* to identify OI state of the research in software engineering. OI has attracted a lot of researchers across different domains. However, it remains unexplored in software engineering. *Paper I* systematically explores the existing OI literature with the focus on software engineering.

RG2 is formulated to understand how the term Product, process, business and organization *Innovation* is perceived and understood among employees of a software-intensive organization. *Paper II* was designed to meet the *RG2* and it highlighted the lack of understanding of the term *innovation* among employees at all levels [95].

RG3 is relevant to investigate OI on a product level in a software-intensive organization and influenced by *RG1*. The literature lacks evidence about the performance of OI on the fined grained product development level [23]. *Paper III* is aimed at exploring why and how a software-intensive organization adopts OI. In addition, *Paper III* also points out the innovation outcomes gained by the case organization.

RG4 leads to Paper IV, focused on exploring the testing activities of OI based project and identifying the key challenges associated with the testing in OI.

Figure 2: Research Overview (See synthesis in section 5)



4 Research methodology

Several research methods were utilized to meet the research goals. The thesis mainly consists of exploratory and evaluative empirical research [146], based on studies using a systematic mapping study [113], survey [46] and case study [122] research method (see Table 2).

Paper I presents a systematic mapping study designed to explore the OI literature on software engineering. Prior reviews were either not systematic [68, 150], partly systematic [142] or, for example, focus on the history or evolution of OSS or available innovation metrics [40]. Moreover, these reviews lack objective quality criteria to support the interpretation of the results to evaluate OI performance. It is to be noted that the main focus of *Paper I* was to explore OI in software-intensive organizations and *not* the use of software to support OI. Furthermore, it was not possible to start with the clear cut research questions due to lack of evidence for a systematic literature review. Therefore, a systematic mapping study was chosen over the systematic literature review in order to explore the OI notion in software engineering.

In *Paper II*, a qualitative survey [94] is presented to understand the perception of innovation, particularly, how product innovation, business innovation, market innovation and process innovation interplay with each other [4]. The main trigger for conducting this survey was the interest of the case company in exploring whether or not employees have a common understanding of the term innovation. A qualitative survey was chosen over interviews to get a more generalized view point from hundreds of employees (229 responses) across the whole organization in the limited amount of time. It was more desirable to draw the conclusion based on 229 responses rather than a limited number of interviews.

Paper III presents a case study, which not only investigates OI in an exploratory manner but also makes an attempt to evaluate OI performance in software-intensive organizations. The research questions in *Paper III* are partly based on the findings from *Paper I*. First, the study explores the top contributors in the development of Gerrit and Jenkins (see Table 1). Second, it explains the transition process from *Closed Innovation* to *Open Innovation*, and the key triggers for the case company towards this transformation. Third, it maps the existing practices of requirements engineering and testing with the identified OI challenges. The study made an attempt to understand how the aforementioned software engineering processes interact in OI. In order to achieve the aims, the study uses the flexible case study design to explore OI in software engineering since it is more suitable for exploratory studies. The quantitative data extracted from the source code repositories is used as a basis for identifying the type of contributions made by the case company and also the key interviewees in the studied units of analysis.

Paper IV idea came up from the interviews conducted in *Paper III* while investigating testing activities in OI. One of the interviewees mentioned the initiation of *Acceptance Test Harness (ATH)*, which helps developers to test Jenkins with automated test cases. As a consequence, *Paper IV* presents a case study conducted to explore if the ATH testing activities adhere to the ISO/IEC/IEEE 29119 testing standard. *Paper IV* compares and contrasts OI testing process with the standard testing process and sets the agenda for future work with the main focus on software testing in OI (see Fig. 2).

The case company used in the studies is Sony Mobile and the units of analy-

sis are Gerrit [3] and Jenkins [2]. Both Jenkins and Gerrit are OSS tools part of Sony's continuous integration tool chain. Sony Mobile is a multinational organization with more than 5,000 employees globally, developing embedded devices. The chosen branch in the thesis is responsible for the development of Android phones. Furthermore, Sony is becoming more and more open in terms of using OSS communities. The transition from the *closed innovation* to *open innovation* could be a good comparison to OI in software engineering. Therefore, Jenkins and Gerrit are OSS examples studied in the thesis seen as an enabler for OI in software engineering.

Table 2: Research strategy [122]

Paper Id	Research Object	Research Strategy
Paper I	Exploratory	Systematic Mapping Study
Paper II	Exploratory	Survey
Paper III	Exploratory and Evaluative	Case Study
Paper IV	Exploratory	Case Study

5 Results and synthesis

This section summarizes the results from the papers included in this thesis. For each paper, we stated the rationale, the methodology used and the key findings.

RG1: To synthesize the research knowledge on OI for software-intensive organization

Paper III identifies 33 studies, divided into nine themes as a result of thematic analysis [31]. 17 out of 33 studies were found to be in high rigor and relevance category and 12 studies were found in the high relevance and low rigor category indicating that results are highly relevant to industry. The key themes identified in the study are as follow:

1. IP strategies
2. OI toolkits
3. Degree of openness
4. OI models/frameworks
5. Managerial implications

6. Enabling OI communities

7. Benefits

8. Challenges

9. OI strategies

Each of the above mentioned themes is defined in detail in *Paper I* with corresponding empirical evidence associated with it. Furthermore, we classified papers based on the research methodology [122] and paper classification [146] followed by the rigor and relevance analysis [71]. 20 evaluation papers used case study research methodology, seven were survey evaluation, two proposal papers each with survey and framework followed by one framework validation and a tool proposal paper.

In conclusion, the results indicate that start-ups have a higher tendency to opt for OI compared to incumbents and firms assimilating knowledge into their R&D activity have a better chance of gaining financial advantage. Furthermore, an important implication for industry is that OSS and OI does not come for free. Software-intensive organizations must invest in the OSS communities with a clear resource investment plan to leverage their key resources. The large share of evaluation research alludes to researchers to produce more solution oriented papers followed by the validation.

RG2: To understand the perception of innovation in a large scale product-focused software organization

Paper II focuses on investigating how innovation is perceived among the employees of the case company. Four innovation types namely, product innovation, process innovation, business innovation, organizational innovation and the interplay between them were studied in a survey at a local branch of a multi-national organization. 229 responses were received including 469 free text comments.

Qualitative data analysis indicated that employees have trouble relating to the term *Innovation*. Employees considered themselves not able to perform or participate in innovation activities since they do not see it as a part of their work. Therefore, there is a need to create the mindset that innovation is possible. The majority of respondents think that product innovation triggers process innovation and business innovation. However, the product and business innovation seems quite complementary. Furthermore, we identified challenges for all four innovation types and it may be implied that the company has challenges in managing OI concepts due to lack of clear understanding of the term innovation.

RG3: To explore and evaluate how a software-intensive organization uses OSS as an enabler for OI and gains benefits

Paper III investigated the OSS tool usage and involvement of Sony Mobile. The units of analysis were Jenkins and Gerrit, the central tools in Sony Mobile's continuous integration process. Moreover, the study also investigated how Sony Mobile captures value externally using OSS communities. We started by extracting the Gerrit and Jenkins change log data to classify Sony Mobile's contributions, and to identify the key contributors for interviews.

The results of the study suggest that moving from *Closed Innovation* to *Open Innovation* model was a paradigm shift around 2010 when Sony Mobile moved from the Symbian platform to Google's open source Android platform in its products, driven bottom-up from the engineers at Sony Mobile. Jenkins and Gerrit are not seen as a competitive advantage or a source of revenue, which indicates that Sony Mobile's openness is limited to the non-proprietary and non-competitive tools only. Furthermore, the requirements process in the Tools department was optimized to work towards the Jenkins and Gerrit communities. The Tools department team works in an agile manner with the influences from Kanban for simpler planning.

On the other hand, the Tools department is struggling to test Gerrit with the old manual testing framework. The openness made the Tools department think of switching from the manual to an automated testing process. Consequently, an Acceptance Test Harness is created to contribute internal acceptance tests to the community and have the community to execute what Sony Mobile tests when setting up a next stable version and vice versa. More so, requirements prioritization and bug fixes are prioritized based on the most pressing needs of Sony Mobile. Paper III further explores if there are any innovation outcomes attached to these tools and identified the following innovation outcomes (see Paper III) as results of these tools in Sony Mobile's continuous integration process:

1. Free features
2. Free maintenance
3. Freed up time
4. Knowledge retention
5. Flexibility
6. Increased turnaround speed
7. Increased quality assurance
8. Improved new product releases and upgrades

9. Inner source initiative

Sony Mobile uses dedicated resources in the Tools department to work with the Jenkins and Gerrit communities. Furthermore, we also discovered that Sony Mobile lacks key performance indicators to measure its innovation capability before and after the introduction of OI in the Tools department. However, the qualitative data suggests that OI results in improved stability and flexibility in the development environment. The findings of the study are limited to software-intensive organizations with the similar domain, size and context as Sony Mobile.

RG4: To investigate how software testing is performed in OI

Paper IV extracts and analyses the change log data of ATH in conjunction with the work on *Paper III* to pinpoint the top contributors to ATH, followed by five semi-structured interviews. It is to be noted that, interviewees were identified from the change log data to know their opinion about the ATH testing and challenges associated with it.

To conclude, the ATH testing process does not strictly follow the ISO/IEC/IEEE testing standard due to the absence of formal test plans. Furthermore, testable objects are derived based on the individual stakeholders needs without consulting other stakeholders in the community. Taking the high number of Jenkins plugins (1000+) into account, it becomes increasingly difficult to have a complete test coverage since the test cases depends on the subjective judgment of developers. *Paper IV* concludes that ATH enables developers to identify problematic areas (plugins) and take corrective actions rapidly whenever there is an update in the Jenkins core or any of its plugin.

Synthesis

This section provides answers to the four research goals by synthesizing the results from the included papers, see Table. 3. The synthesized evidence from *Paper I* suggests that start-ups have a higher tendency to adopt OI compared to incumbents since start-ups engage themselves in OSS to quickly acquire the knowledge. OI provides some initial gains for smaller companies but also has the tendency to limit the organizational performance when the level of participation increases above average [70, 131]. In order to gain financial benefits, it is imperative for an organization to have high the absorptive capacity to utilize the available technical knowledge [32]. Business strategies play a vital role in embracing OI and companies may pursue their differentiation strategy with a controlled degree of openness. However, it needs to be explored which strategy is optimal, given the company's size domain and product characteristics as mentioned by Mowery [104]. Furthermore, adopting OI seems rather like a reactive strategy than a proactive

strategy to hold on to an organizational competitive advantage. An interviewee mentioned in *Paper III* that the transition from *Closed innovation* to *Open Innovation* was made from existing proprietary solutions to the tools used by Google in their Android development to get as close as possible to Google's tool chain. This transition process was driven bottom up with the support of management to ease off the old and complex chain of integration and building solutions.

Regarding, determinants of openness, several factors interplay in the decision making process whether or not a new feature or a project should be open. Evidence suggests (*Paper III*) that openness is limited to the tools that are not the direct source of revenue (commodity). However, it does have an indirect impact on the propriety product development labeled as innovation outcomes in *Paper III*. OI impacts on product development by increasing the turnaround speed for new releases and upgrades, better quality assurance and frees up the time for more innovative activities. In addition, interviewees in *Paper III* acknowledged that it is not only the free work that motivates organizations but also the risk of lagging behind by not choosing to work with communities.

Table 3: Research Synthesis

Paper I	Paper II	Paper III	Paper IV
• OI remains unexplored in SE • Lack of systematic effort to summarize OI in SE • Nine distinct themes identified: 1) IP strategies, 2) challenges, 3) Benefits, 4) OI Communities, 5) Managerial implications, 6) OI toolkits, 7) degree of openness, 8)OI instruments/strategies, 9) OI models/frameworks • 29 out of 33 studies were reported with high relevance • Results are relevant to industry due high relevance • OI should be used in a complementary role instead of a replacement to accelerate the internal innovation	• Innovation is promoted to stay competitive • Product innovation triggers process and business innovation • Product and organizational innovation are complementary • Increasing awareness about the four innovation types (Product, process, business, organizational) may improve the innovation • Objective interpretation of innovation types needs to be promoted	• Adopting OI was a paradigm shift and essentially driven bottom up limited to the non competitive tools • Opening up is becoming more and more acceptable at Sony Mobile • RE process is informal and prioritization is based on the individual needs of stakeholders • Ongoing transition from a manual testing to automated testing using Acceptance Test Harness • Freed-up time, increased speed, free features, inner source initiatives, Knowledge retention, increased quality assurance are some of the key innovation outcomes found as a consequence of OI	• ATH process does not strictly adheres to ISO/IEC/IEEE due to independent identification of testable features from stakeholders without any formal test plan • Difficult to attain the complete test coverage due to 1000+ plugins in Jenkins • Defects detection process of Jenkins became faster because the developers can quickly run test cases on their local branch to detect the introduction of bugs before merging the code with the main branch

Based on Conboy [27], the interplay between OI and agile where Agile seems to create barriers in transferring the ideas outside the team boundaries, mainly due to short iterations, stand-up meetings, limited documentation, and features backlog reduces the time for trying out new things or sharing ideas outside your team.

Furthermore, managers can enhance the firm's innovativeness by encouraging their employees to participate in communities. Consequently, the Tools dept. at Sony Mobile is spearheading the culture of being active or in engaging with communities. However, they have too few resources in terms of time and budget to make a significant contribution in the OSS communities. The key implication for managers is that the organization may adopt differentiation strategies to achieve openness without isolating their developers from the community. There is a risk that other software companies may not devote their best employees to work in the community or may only passively participate in the development process. Therefore, too open behavior might be commercially harmful for the organization.

Software-intensive organizations promotes innovativeness among their employees in order to keep up with the competitive nature of ever growing technology, but the meaning of the term innovation and the interplay between organizational, business, product and process innovation are not well understood across different units of the studied organization (see *Paper II*). An interviewee in *Paper II* stated, *"...I recognize that [company] does this often [...] But I am not sure if it is really innovative or just mindless changes."*. The borderline between when something goes from being an improvement or common functionality to an innovation is subtle. In addition, some employees believed that it is not the part of their work description or role. A tester stated in *Paper II* that working with testing does not lead much improvement in the product besides some ideas that pops up occasionally. Therefore, there is a need to increase awareness and knowledge of different innovation types with clear role descriptions which may improve the innovation in an organization.

Building on to the testing part, Jenkins and Gerrit communities both focused on manual test cases. The openness led Sony Mobile to think about automated testing for Jenkins launching the *Acceptance Test Harness* to improve quality assurance. It may reduce the work load internally and may secure that the settings and cases specific to Sony Mobile are taken care of. On the other hand, the community gets the view and settings from a large company which enables community development.

However, when it comes to requirements and bugs prioritization , all stakeholders prioritize their most pressing needs. An interviewee surveyed in *Paper III* emphasized that they almost never implemented any feature requests from outside unless Sony Mobile thought that it is a good idea. To summarize, OI became more and more accepted practice in the Sony Mobile with the passage of time due to the associated innovation outcomes. In effect, OI makes internal development faster.

6 Ethical aspects and threats to validity

The investigation of OI at the case company was triggered as a result of difference of opinion on OI between engineers, middle management and higher management. Engineers believe that opening up development process can give them a cutting edge over their competitors. By doing so, they can steer the community towards their business model and make the community work for them by contributing their code back to the community. However, the open strategy required some time to be accepted. The results of the *Paper III* have the potential to influence the top management decisions regarding OI adoption. A typical example could be to present negative findings to discourage OI initiatives. Therefore, it is vital for researchers to present the findings as objectively as possible.

OI research in SE involves software engineers working in the industry. The investigation started from mining the OSS code repositories to identify the key contributors and possibly classify their contributions in terms of new features, bug fixes, cosmetic issues or documentation. However, the data collection process about research subjects and the case company is a two step process:

1. Collect the public data about software engineers and the case company from GitHub

2. Conduct interviews based on data collected from the code repositories

It is worth mentioning that the case company has shown a strong interest in investigating its OI activities to see whether or not it is helping them to accelerate their internal innovation process. Sony Mobile gets recommendation whether or not opening up in their development process gives them a cutting edge on their competitors and the researchers are able to publish research papers to carry forward OI state of the art in software engineering. Therefore, its a collaboration that leads to a win-win situation for both stakeholders. On the hind side, there are risks attached to the research process. Specifically, the case company fully understands the importance of collaboration with the research community and its positive impacts on their internal processes of working. However, if a local newspaper correspondence decides to picks up something (e.g. internal conflicts) randomly from the study out of the context and place it on the front page of the local newspaper may lead to a massive dent on concerned organization's reputation.

Apart from ethical aspects there are validity concerns that worth mentioning about the thesis. Internal validity is the confidence that we can place in the cause and effect relationship in a scientific study [122]. In the thesis, review protocols were created for all the studies and reviewed by all authors to be more objective and to assure quality as well. The studies revealed that Sony Mobile does not have any metrics to measure innovation thus, researchers had to rely on implications drawn from qualitative data collected from interviews. The element of subjectivity was addressed by performing the analysis independently by multiple researchers.

External validity refers to ability to generalize the study findings [122]. In particular, all those software-intensive organizations using Jenkins and Gerrit in their continuous integration process have the context similar to Sony Mobile. Therefore, the findings of the studies included in the thesis may be generalizable to organizations using Jenkins and Gerrit in their continuous integration process.

Construct validity refers to choosing the right measure for the concept under study [122]. One of the deficiencies found in the literature was that OI investigated at the firms level to see its innovative impact, which leads to the introduction of confounding factors. Therefore, in this thesis we aimed at investigating OI at the project level to eradicate confounding factors that may or may not lead to firms innovative performance. More so, Gerrit and Jenkins (see *Paper III*, and *Paper IV*) are not the typical examples of revenue producing softwares but can be seen as a good candidate for investigating OI using OSS at a project level and its impact on Sony Mobile's innovative performance.

Reliability deals with the ability to replicate the same study with the same results [122]. To address the reliability concerns, review protocols, multiple data sources, independent qualitative and quantitative data, and interview transcription summary validation by interviewees were some of the techniques used in the studies to draw conclusions more reliably. Finally, the study design and findings of the studies were kept transparent in terms of mentioning the context of case company except for the anonymous interviewees names.

7 Future work

As a next step after licentiate, the plan is to move from exploratory studies to evaluation and solution oriented work. The following research goals are formulated for future research work.

RG1 Validate the findings of *Paper III* across the software-intensive organizations and write more generalized results for practitioners regarding adoption of OI.

RG2 Investigate the contribution strategy of the case company in relation to the product complexity. More specifically, we want to see if contributing to the community would save the case company time in relation to a patching strategy.

RG3 Investigate the testing challenges experienced by developers in OI enabled projects as opposed to closed source projects.

In pursuit of aforementioned goals, three studies are planned to further investigate OI in software engineering (see Fig. 2). The research plan for the three studies is as follows:

Study V is planned to investigate OI trends in Swedish software industry. We plan to design a survey as an extension to *Paper III* to generalize the findings and possibly write recommendation for practitioners regarding when, how and why to adopt OI in SE. The survey questions will be divided in the following categories:

1. Demographics
2. Involvement in OI using OSS projects (Why)
3. Requirements engineering process and OI (How)
4. Testing process and OI (How)
5. Business strategy (When)
6. How OI is measured in Software Engineering?

This study will give us more information on OI performed on a product development across the software-intensive organizations. As a result, it should be possible to generalize the answers to the questions, particularly Why, how and when, on the whole Swedish software industry.

Study VI is planned to investigating Sony Mobile's contribution-strategy to leverage their resources in an efficient manner. Contribution of certain bug fixes and feature implementations without a thoughtful process may implicate the giving away of differentiating features, or doing something that is already a commodity. Therefore, ***Study VI*** is aimed using Kraljic Matrix [83], a tool to evaluate the OSS contribution strategy in terms of its business impact in relation to control complexity. Furthermore, we plan to utilize the three-layer product model for managing system growth, suggested by Bosch [15]. The idea is to further evaluate Sony Mobile's contribution strategy based on the following three layers:

1. Commoditized functionality
2. Differentiating functionally
3. Innovative and experimental functionality

Study VII is planned to get the overview of issues experienced by testers in OI enabled projects. Limited information is available in the literature regarding the negative experiences of developers with automated testing. The study conducted by Wiklund [147] analyzed and classified a company's issues in the internal discussion board for test automation tools. One of the key reasons for using a discussion board instead of a ticketing system is that the discussions are more transparent and provide an opportunity for building a future knowledge bases for automated testing support. However, findings suggest that only 32% of the users considered forums to be a viable support alternative and users expecting a quick response are less likely to use forums. Instead, framework developers alluded that they get many

request for support through emails, telephone and instant messages. Therefore, it would be interesting to compare the *Closed Innovation* test automation with *Open Innovation* test automation process to better understand testing in OI context.

The overall goal is to build on to the exploratory part of the research presented in the licentiate thesis and move towards solution oriented OI studies. Particularly, we want to write more generalized guidelines for researchers and practitioners based on the future studies. These guidelines will entail when and how to adopt OI and how open software-intensive organizations should adapt their processes (i.e testing) to maximize their innovative performance. The possible collaboration for *Study VII*, *Study V* and *Study VI* may include professional software engineers, testers and managers actively contributing to OSS communities, and working in software-intensive organizations.

INCLUDED PAPERS

CHAPTER I

OPEN INNOVATION IN SOFTWARE ENGINEERING: A SYSTEMATIC MAPPING STUDY

Abstract

Context: Open innovation (OI) means that innovation is fostered by using both external and internal influences in the innovation process. In software engineering (SE), OI has existed for decades, while we currently see a faster and broader move towards OI in SE. We therefore survey research on how OI takes place and contributes to innovation in SE.
Objective: This study aims to synthesize the research knowledge on OI in the SE domain.
Method: We launched a systematic mapping study and conducted a thematic analysis of the results. Moreover, we analyzed the strength of the evidence in the light of a rigor and relevance assessment of the research.
Results: We identified 33 publications, divided into 9 themes related to OI. 17/33 studies fall in the high–rigor/high–relevance category, suggesting the results are highly industry relevant. The research indicates that start-ups have higher tendency to opt for OI compared to incumbents. The evidence also suggests that firms assimilating knowledge into their internal R&D activities, have higher likelihood of gaining financial advantages.
Conclusion: We concluded that OI should be adopted as a complementary approach to facilitate internal innovation and not to substitute it. Further research is advised on situated OI strategies and the interplay between OI and agile practices.

1 Introduction

Open innovation (OI) and associated free exchange of information about new technologies are recognized as one of the main drivers for collective inventions in the 19th century by Allen [7]. Two decades after Allen's paper from 1983, Chesbrough's seminal book about OI [24] has initiated an unabated interest [51] among researchers in innovation management [68], economics, psychology, sociology, and also Software Engineering (SE) [135]. The work initiated by Chesbrough [24] forced both practitioners and scholars to rethink the design of innovation strategies in a networked environment [68]. The inherent flexibility of software, combined with increase of software cost and value for new products and services, puts SE into the hotspot of OI. Several trends, such as outsourcing, crowd-sourcing and funding, global software development, open source software, agility, and flexibility, challenged the *do it yourself* mentality [49]. More courageous voices suggested even that closed innovation might have been the exception in the history, characterized mostly by open innovation practices [104].

OI is a relatively new field of research and a collective theoretical foundation is starting to emerge. Chesbrough [24] was the first to define OI as "*a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology*". OI encompasses various activities such as inbound, outbound and coupled activities [50], and each of these activities can be more or less open. Open Source Software (OSS) is the most straightforward application of OI to software development [68], although not the only one [150]. The success of OSS in the last twenty years have ignited and encouraged several new movements for collective innovation such as: outsourcing, global software development, crowd-sourcing and founding.

Despite the wide interest in several domains and the unquestionable potential that OI can bring to the software industry, OI remains greatly unexplored in the SE literature, while in the OI literature extensive interest is given to exploring OSS as one of the ways to incorporate external knowledge and innovation to internal product innovation [24]. Similarly in the early days of OSS, many interesting OI initiatives were performed, e.g. opening up software product organizations and utilizing open configurations [73]. However, there is a lack of systematic efforts that focus on summarizing the current state of the literature on the relation between OI and SE. Previous reviews are either not systematic [68, 150], partly systematic [142] or, for example, focus on the history or evolution of OSS or available innovation metrics [40]. Moreover, these reviews lack quality criteria to support the interpretation of the results in favor or against OI.

Therefore, we identified a need to systematically review OI research in SE with a specific focus on assessing the strength of the empirical evidence in the identified studies [71], highlighting the current themes and outlining implications for research and practice. For instance, a study might have high relevance (e.g.

managerial implications for an industrial scale project), but at the same time have low rigor (e.g. having validity threats and lacking descriptions of the units of analysis). Consequently, these above mentioned needs lay the foundation for a systematic mapping study [113] to explore the concept of OI in the context of SE. Specifically, this mapping study makes the following contributions:

1. Identification of the existing themes and patterns in the literature for open innovation in software engineering.

2. Assessment of trustworthiness of the results with respect to rigor and relevance [71].

3. Based thereon, identification of knowledge that may inform industry practice on open innovation in software engineering

4. Identification of the research gaps for further exploration of open innovation in software engineering [78].

The remainder of the paper is structured as follows: Section 2 presents related work and Section 3 presents the research method (review protocol). Next, Section 4 highlights the results of the search and the analysis the synthesized research, followed by a discussion in Section 6 which results in a research agenda and advice for industry practice in Section 6. Section 7 concludes the paper.

2 Related work

Using the study by West and Bogers [142], we identified four secondary studies (literature reviews) on OI [40, 68, 142, 150], relevant to this study. The studies are summarized in Table 1.

Are the reviews systematic? Huizingh [68] and Wnuk and Runeson [150] conducted reviews on OI, however neither of them is systematic according to the guidelines stated by Kitchenham et al. [77]. The study conducted by West and Bogers [142] could be considered partly systematic, since the relevance can be seen in terms of data sources, inclusion/exclusion criteria and data extraction. On the other hand, the review conducted by Edison et al. [40] adheres to guidelines by Kitchenham et al. [77] and Petersen at al. [113]. In this paper, we report a review conducted according to the guidelines by Kitchenham et al. [77].

What were the objectives behind conducting reviews? West and Bogers [142] conducted a review on OI with the main objective to define an agenda for OI research. They classified the studies into three main categories of OI, namely, inbound (outside in), outbound (inside out) and coupled, as suggested by Enkel at al. [42]. Wnuk and Runeson [150] performed a study with the goal to propose a SE framework for OI.

Huizingh [68] also focused on exploring the notion of open innovation and on the degree of OI adoption by the firms. The study concluded that the knowledge about *how* to apply OI and *when* to do it is still incomplete. Edison at al. [40] centered their literature study around innovation measurement and innovation management aspects, e.g. definitions, frameworks and metrics. Our study limits its scope to SE and focuses on deriving existing OI themes and patterns using thematic analysis. Moreover, this study also focuses on exploring the strength of evidence under the light of rigor and relevance, and states the further course of action in terms of OI in SE.

What were the data sources used in the reviews? Were the used search terms appropriate? Huizingh [68] neither specified the database, nor the search terms used. Likewise, West and Bogers [142] did not mention the search terms for their study, but provided the time scope of the survey (between 2003 and 2010) and the list of selected management journals, see Table 1. Conversely, the study conducted by Wnuk and Runeson [150] used Inspec and Compendex and the following search terms "Open innovation, requirements engineering, testing, software and methodology". However, the time span for the search is not reported. Edison et al. [40] used multiple data sources namely, Inspec and Compendex, Scopus, IEEE explore, ACM digital library, Science direct, Business Source Premier (BSP) and performed the search between 1949 and 2010, see Table 1. Their search terms aim at identifying innovation metrics, measurements, drivers and innovation attributes. Inspired by the previous reviews, we organized our search string into three main categories and employed the inclusion exclusion criteria after the search process, with keywords: i) related to OI, ii) on SE in order to restrict the results to the SE domain, and iii) pertaining to empirical evidence on OI (see Section 3.3). Furthermore, we complemented our search string with backward snowball sampling [72, 123] by scanning the reference list of all primary studies, see Section 3.2.

Did the reviews use any quality assessment criteria for primary studies before analyzing their results? Neither Wnuk and Runeson [150] nor Huizingh [68] used explicit quality assessment criteria for the identified studies. On the other hand, West and Bogers [142] included studies that focused on OI as per the definition by Chesbrough [24] and excluded book reviews, commentaries and editorial introductions. Edison et al. [40] used a set of questions for quality assessment and to evaluate if a study explains the aims, methodology and validity threats. We used a comprehensive set of guidelines that cover rigor and relevance of studies. We slightly tailored the criteria from Ivarsson et al. [71] to fit into the scope of this study, see Section 3.4.

How did the reviews extracted data from primary studies? Did they map data extraction with research questions? The data extraction strategy was not reported in three studies [68, 142, 150]. The information about the mapping between the data extraction properties and the research questions was also absent. However, Edison at al. [40] described the data extraction strategy which was piloted before

the execution to ensure a common understanding among all involved researchers. We created a defined set of data extraction properties, and mapped them on research questions to avoid redundant information, outlined in Table 3.

How did the reviews synthesize the data from primary studies? Neither of the four studies followed an established procedure for the synthesis, such as thematic or cross-case analysis [30, 31]. Instead, West and Bogers [142] used a self created four phase integrated model (i.e. obtaining, integrating, commercializing, interaction with communities) to guide the literature review and classified studies based on dimensions provided by Enkel at al. [42]. Similarly, Wnuk and Runeson [150] presented the synthesis in a table where studies are categorized in terms of research type (e.g. evaluation, proposal, opinion, solution, conceptual etc.) defined by Wieringa et al. [146]. Moreover, studies were also classified in terms of software techniques, process and methods, and presented a framework to foster OI with technical and methodological dimensions stated above.

Edison et al. [40] presented their synthesis in terms of different types of innovation definitions available in the literature, metrics used to measure innovation, and challenges related to existing innovation measurements. They developed a model to assist organizations to use the available measures to develop insights into their innovation program. Finally, Huizingh [68] wrote a literature review without synthesis.

In summary, this systematic study aims at exploring the OI in SE in a much more rigorous manner according to guidelines of Kitchenham et al. and Petersen et al. [77, 113] and focusing on systematic synthesis of the findings.

Facets	West and Bogers [142] (2013)	Edison et al. [40] (2013)	Wnuk et al. [150] (2013)	Huizingh [68] (2010)
Systematic	Partly	Yes	No	No
Data sources	1. Academy of Management Journal 2. Academy of Management Review 3. Administrative Science Quarterly 4. California Management Review 5. Harvard Business Review 6. IEEE Transactions on Engineering Management 7. Industrial and Corporate Change 8. International Journal of Technology Management 9. Journal of Product Innovation Management 10. Long Range Planning 11. Management Science 12. MIT Sloan Management Review 13. Organization Science 14. R&D Management 15. Research Policy 16. Research-Technology Management 17. Strategic Management Journal 18. Technological Forecasting and Social Change 19. Technovation	1. Inspec and Compendex 2. Scopus 3. IEEE Xplore 4. ACM Digital Library 5. ScienceDirect 6. Business Source Premier (BSP)	Inspec and Compendex	N/A

Facets	West and Bogers [142] (2013)	Edison et al. [40] (2013)	Wnuk et al. [150] (2013)	Huizingh [68] (2010)
Repeatability	No	Yes	No	No
Quality Assessment	No	Yes	No	No
Inclusion/exclusion criteria	Partly	Yes	No	No
Data extraction properties	Partly	Yes	No	No
Validation of results	No	Yes (Piloted the criteria)	No	No
Data synthesis	Partly (without mentioning its type)	Yes (without mentioning its type)	Exploration instead of synthesis	No (Only conclusion)
Purpose	An attempt to define an agenda for open innovation research	This study explores various aspects relevant to innovation measurement ranging from definitions, measurement frameworks and metrics that have been proposed in literature and used in practice	This paper proposes a software engineering framework, designed to foster open innovation by designing and tailoring appropriate software engineering methods and tools.	The study intends to explore the challenges that both practitioners and academics face in understanding the open innovation concept. The paper also focuses on less understood areas of open innovation that "require management attention and offer fruitful ideas for further academic research".
Outcome	The study defined the agenda for open innovation research and concludes with recommendations for future research that include examining the end-to-end innovation commercialization process, and studying the moderators and limits of leveraging external sources of innovation	A systematic literature review followed by an online questionnaire and interviews with practitioners and academics were employed to identify a comprehensive definition of innovation that can be used in software industry. The metrics for the evaluation of determinants, inputs, outputs and performance were also aggregated and categorized. Based on these findings, a conceptual model of the key measurable elements of innovation was constructed from the findings of the systematic review.	This study discusses the methodological and process dimensions and outlines challenge areas that should be reviewed when transitioning to software engineering driven open innovation.	The study shows that open innovation has been a valuable concept for so many firms and in so many contexts in innovation management. However, the knowledge about *how* to do it and *when* to do it remain dispersed and incomplete.

Table 1: Summary of existing literature reviews

3 Research methodology

In this section, we present the literature review methodology, based on the guidelines provided by Kitchenham et al. [77] and Petersen et al. [113]. The study was conducted in six steps outlined in subsections below: I) identification of primary studies, II) search string development and database search, III) performing including and exclusion criteria, IV) data extraction, V) quality assessment through rigor and relevance, and VI) synthesis and reporting.

3.1 Research questions

The research questions for the mapping study are defined as:

RQ1: Which themes and patterns of OI in SE exist in the literature?

RQ2: How strong is the evidence in favor of or against OI in SE?

3.2 Identification of primary studies

In order to identify the primary studies, following steps were performed, see Figure 1.

1. Identification of 15 control papers [35] from forward snowball sampling [54, 72].

2. Extraction of studies from databases using a search string: 2805 papers were identified using a search string

3. Duplicate elimination at the database level: 305 studies were found to be duplicates, and hence removed.

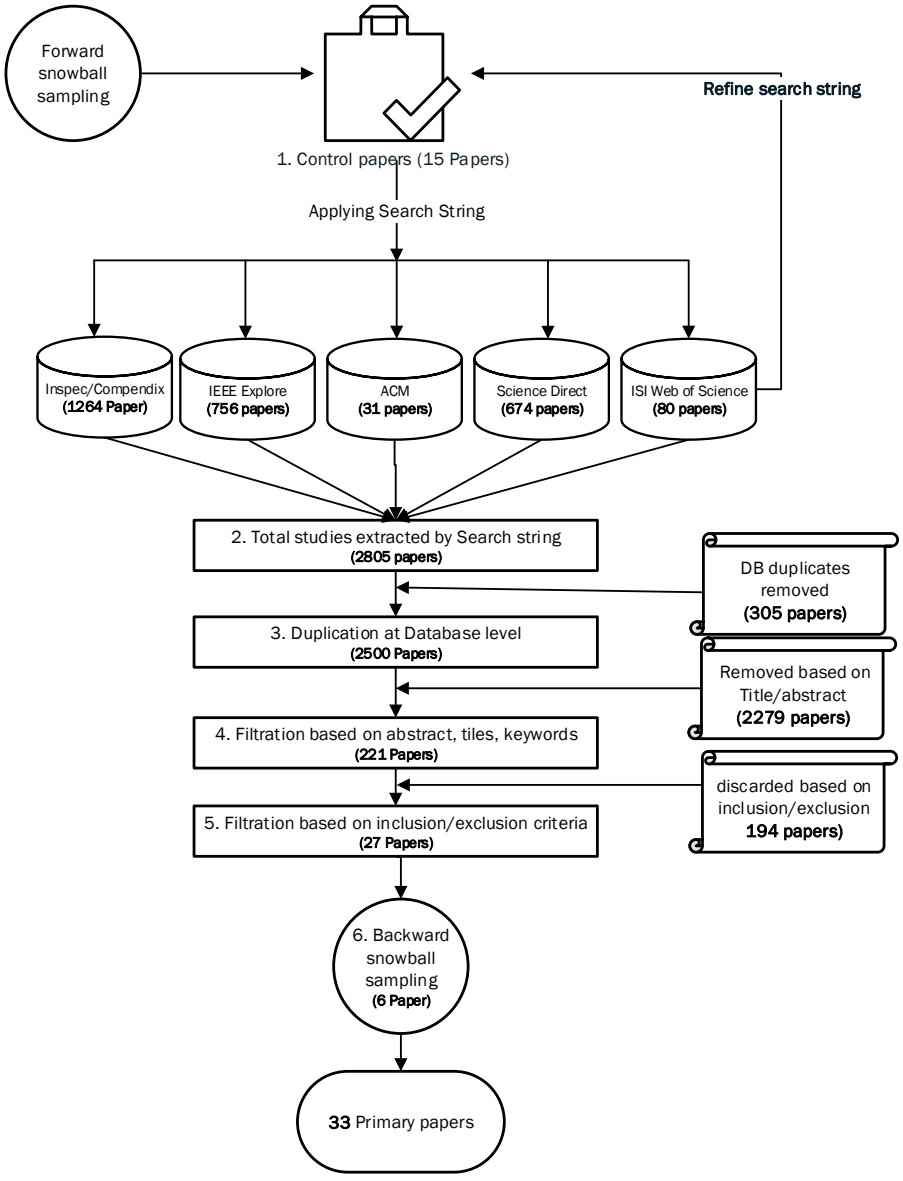
4. Selection of studies based on abstract, titles and keywords: 2279 papers were not found relevant and excluded.

5. Filtering based on inclusion/exclusion criteria: 194 additional papers were excluded after applying the inclusion/exclusion criteria and 27 papers were found to be relevant and pertain to the scope of this study

6. Backward snowball sampling was applied to scan the reference list of 27 primary papers and enabled us to spot 6 more relevant papers.

We identified 33 studies that directly pertain to the scope of the study. The additional studies found by the snowball sampling confirms the usefulness of snowballing for identification of potential studies missed by database searches.

Figure 1: Identification of primary studies



3.3 Search string strategy

In order to develop the search string, the keywords were aptly derived from 15 control papers, see Figure 1. The search terms are organized into three interventions: T1 includes terms related to open innovation, T2 related to outcomes, T3 related to the research methods.

1. **T1:** Open Innovation OR Open-Innovation OR OI OR innovation OR innovation management

2. **T2:** software OR software ecosystem OR product line OR requirement* engineer* OR requirement* management OR open source

3. **T3:** exploratory study OR lesson* learn* OR challenge* OR guideline* OR Empirical investigation OR case study OR survey OR literature study OR literature review OR interview* OR experiment* OR questionnaire OR observation* OR quantitative study OR factor*

The interventions are combined using Boolean operators (**T1 AND T2 AND T3**) to achieve the desired outcome. We searched the following databases, using their command interfaces and utilizing expert or advanced search capabilities (the search strings used per database are reported in Appendix B):

1. ISI Web of Science

2. Inspec and Compendix (Engineering Village)

3. ACM Digital Library

4. IEEE Xplore

5. Science Direct (Elsevier)

The search string was refined, using the control papers as a benchmark, until the average acceptable level of precision and recall was achieved. A study conducted by Beyer and Wright [12] reported that the recall of the search strategies ranged from 0% to 87%, and precision from 0% to 14.3%. The final search string retrieved 13 out of 15 control papers which gives recall of 86.66%. The final search string achieved precision of 0.52% (13 out of 2500 papers, excluding duplicates). Both precision and recall scores are in range with the findings of Beyer and Wright [12]. The fact that two of the control papers were not captured by the final search string confirms the observations by Wohlin et al. [152] that using single search strategies leads to missing studies. Therefore, we combined database searches with snowball sampling.

3.4 Inclusion/exclusion criteria

The inclusion/exclusion criteria were derived and piloted. These criteria were applied simultaneously on studies to make sure we only include studies that pertains to SE domain and not, for example economics, management or psychology.

Table 2: Inclusion exclusion criteria

Inclusion Criteria (All must apply)	Exclusion Criteria (Each apply separately)
• Peer reviewed papers, and in case of duplicate publications, the priority follows the sequence: Journals, Conferences, Workshops • The study must be accessible in full text. • The study highlights the research-focused concept of OI in the context of software engineering. • The study that reports the benefits, disadvantages, limiting factors, and challenges of OI. • The studies pertaining to the scope of open source software used as OI examples • Factors limiting the adoption of OI in SE • Available tools used by the software community to support OI in SE • Studies that discusses the openness of software producing organization(SPO) • All studies from 1969 to 2013	• All gray and white literature • Non-English articles • Studies about OI in the management and economics context • Intellectual property rights papers • Research on OI not related to SE • All papers that mentioned only the use of software to bring innovation in the fields other than SE. • All articles, which are not within the field of SE in terms of how to develop software • All duplicate studies

The selection of studies was accomplished independently by the two first authors, applying the inclusion/exclusion criteria. In case of uncertainty, the authors included the papers to next step in order to reduce the risk of excluding the relevant papers as suggested by Petersen and Bin Ali [112]. Kappa statistics [84] was calculated at multiple steps in order to check the agreement level between the authors.

First, the Kappa coefficient was calculated on a 10% randomly selected sample of titles and abstracts and it was found to be 0.37. After discussing and resolving the disagreements, the Kappa value increased to 0.91. Second, Kappa was calculated on a sample of randomly selected 50% of papers included into the full text reading phase while applying inclusion/exclusion criteria. Disagreements were identified as the Kappa value (0.48) was found to be below the *substantial agreement* range. Consequently, after discussing and resolving disagreements [112], the kappa value increased to 0.95. It is to be noted that the inclusion/exclusion criteria was applied simultaneously. However, for exclusion it is enough when one exclusion criterion holds.

3.5 Data extraction and synthesis strategy

The data extraction properties outlined in Table 3 were discussed and finalized beforehand. Moreover, a spreadsheet was created for the data extraction properties and also mapped to research questions, see Table 3. The first author performed the data extraction, supervised by the second and the third authors.

The extracted data was synthesized by performed thematic analysis based on the guidelines by Cruzes et al. [31]. First, we identified patterns in the data and then grouped those patterns into distinct themes. Second, in order to check the trustworthiness of each paper, we used rigor and relevance criteria which helped us identifying whether or not results are generalizable to the software industry, see Section 3.6.

3.6 Quality assessment with respect to rigor and relevance

We used the rigor and relevance assessment checklist by Ivarsson et al. [71]. Two researchers reviewed the ratings and data extraction to ensure objectivity. Each paper was assigned a score using objective criteria tailored for this mapping study, see Appendices 1 and 2. The idea behind investigating rigor and relevance resembles the use of a rubric based evaluation in education [71]. Previous studies [74, 103] have shown that rubrics increase the reliability of assessments in terms of inter-rater agreement between researchers.

Rigor can be defined as *"the research methodology is carried out in accordance with corresponding best practices"* [71]. Ivarsson et al. [71] state that rigor has two dimensions: following the complete reporting of the study, and best practices. Through aggregating of study presentation aspects from existing literature, they defined rigor as the degree to which study context (C), design (D), and validity threats (V) are described. All facets are rated on a scale, i.e. weak, medium, and strong description, see Appendix 1.

Relevance deals with the impact of a study on industry [71]. It consists of manifold aspects, namely, relevance of the topic studied [128], ability to apply

Table 3: The data extraction properties explained and mapped to the research question

Category	Properties	RQ Mapping
General information	Authors, Title, Year of Publication, Abstract	RQ1, RQ2
Study Type	Evaluation research, Solution research, Validation research, Proposal research	RQ1, RQ2
Research Methods	Case study, Tool proposal, Survey, Framework	RQ1, RQ2
Research Problem	Description of research questions	RQ1, RQ2
Outcomes	Benefits, limitation, strategies, patterns related to OI	RQ1
Context	Subjects Type (Students/ professionals/researchers/mixed), number of subjects, case description, validity threats to context.	RQ2

a solution in a real world industrial setting with degree of success [153], use of research methods that facilitate industrial realism [129], and provision of a realistic situation in terms of users, scale, and context [71]. We followed the suggestion of Ivarsson et al. [71] to decompose rigor into: users/subjects (U), scale (S), research methodology (RM), and context (C), see Appendix 2.

3.7 Validity threats

This section highlights the validity threats associated with the systematic mapping and how they were addressed prior to the study in order to reduce their impact [122].

Internal validity

The key idea behind conducting the systematic mapping study was to capture available literature as much as possible without introducing any researcher bias thereby, internal validity seem to be a major challenge for the study. In order to address the internal validity concerns, a review protocol was created beforehand and evaluated by three researchers, which took on roles of quality assurance as well. The internal validity is enhanced by following the systematic mapping guidelines [113] and the guidelines for quality assessment criteria [71].

Construct Validity

Construct validity refers to the presence of potential confounding factors and whether or not a study was able to capture what was intended in terms of aims and objectives. One important concern for this study was the multiple definitions of OI. In order to minimize this threat and build on solid foundation, Chesbrough's concept of OI is adopted [24].

External validity

External validity refers to the ability to generalize the results to different settings, situation and groups. The majority of the studies fall into the case study category with high rigor and relevance, see Figure 6. Moreover, many studies were conducted in industrial contexts hence, the results are more general and industry relevant.

Reliability

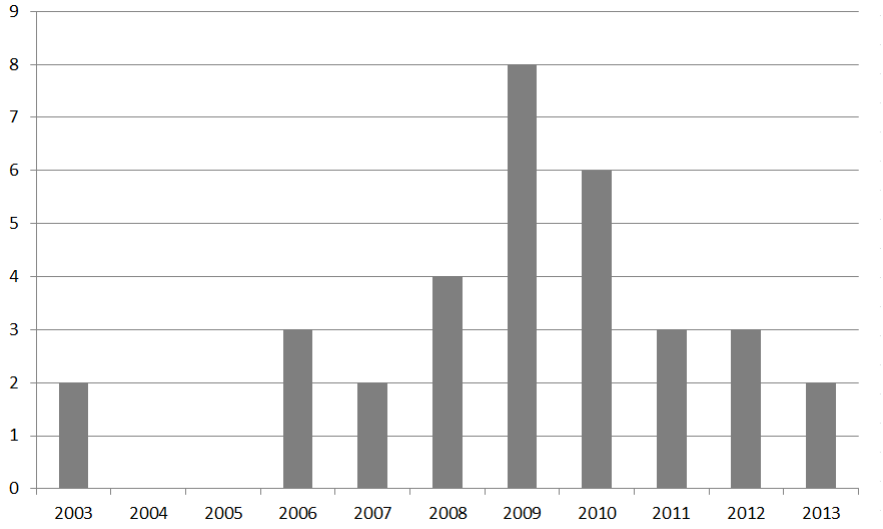
Reliability is concerned with to what extent the data and the analysis are dependent on a specific researcher. Multiple strategies were taken into account in order to enhance reliability. First, there is always a risk of missing out on primary studies with a single search string for all selected databases. Therefore, 15 control papers were identified through forward snowball sampling to verify the precision and recall of the search string. However, this only minimizes the selection bias that may impact further research steps. We believe that the potential effect of this bias have a lesser importance in mapping studies than in SLRs. To further substantiate the search process, backward snowball sampling was applied and resulted in additional studies pertaining to the context of OI in software engineering (see Figure 1).

Second, quality assessment of the identified studies is sensitive on interpretation. Therefore, rigor and relevance criteria were applied to increase the objectivity of this step. The evaluation was performed by the first author and reviewed by the remaining authors. Moreover, we created a data spread sheet and mapped research questions with the data extraction properties in order to comply with the objectives of this study. Besides, all studies were rated according to the rigor and relevance criteria tailored from Ivarsson et al. [71] and data extraction properties from each paper were reviewed by two researchers in the study.

4 Results and analysis

In this section, the results of the mapping study analysis are reported. We give an overview of the time distribution and categorize the studies based on research

Figure 2: Distribution of studies over publication years



methodology used. An analysis of the themes studied is reported, followed by a detailed description of each theme.

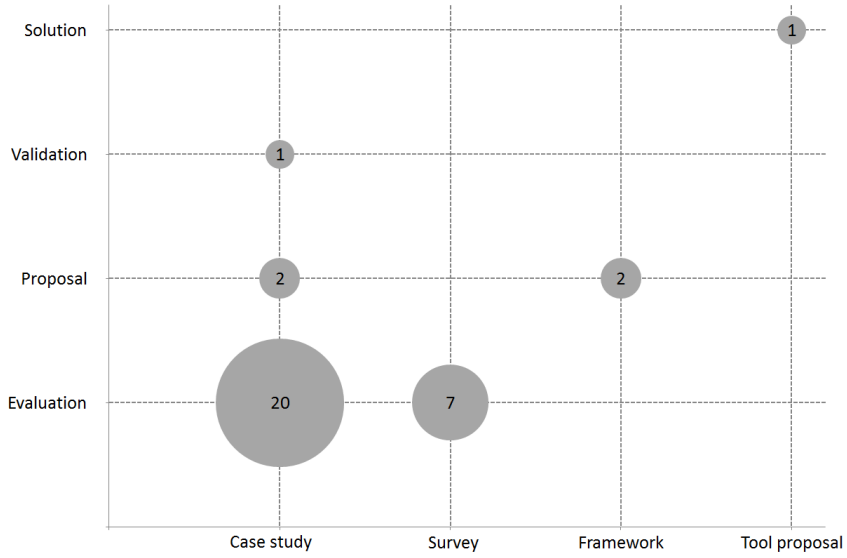
4.1 Distribution of OI studies

33 primary studies about OI in software engineering were found, distributed by their publication year in Figure 2. The scholarly interest in OI seems to be growing at a steady pace since its introduction in 2003 with a maximum annual rate of 8 studies published in 2009. However, the trend declines after that, and it it hard to assess why, since the interest in OI seem to grow in general [68].

4.2 Categorization based on research methodology

Primary studies found are categorized into the research methodology (i.e. case study, experiment, survey etc) and type of the study (i.e. evaluative, proposal, solution, opinion etc) dimensions. The horizontal axis in Figure 3 represents research methodologies defined by Runeson et al. [122] and vertical axis represents the classification of studies established by Wieringa et al. [146]. Evaluations, using case study research methodology dominate among the identified papers with 20 papers, among which two were interview studies that we consider qualitative case studies. Evaluations, using survey research methodology was found in 7 papers. We classified only 2 papers in each of the framework–proposal and case

Figure 3: Research methodology classification based on Runeson et al. and Wieringa et al. [122, 146]



study–proposal categories. Finally, the categories case study–validation and tool proposal–solution received only 1 paper each and no papers were identified in the case study solution category.

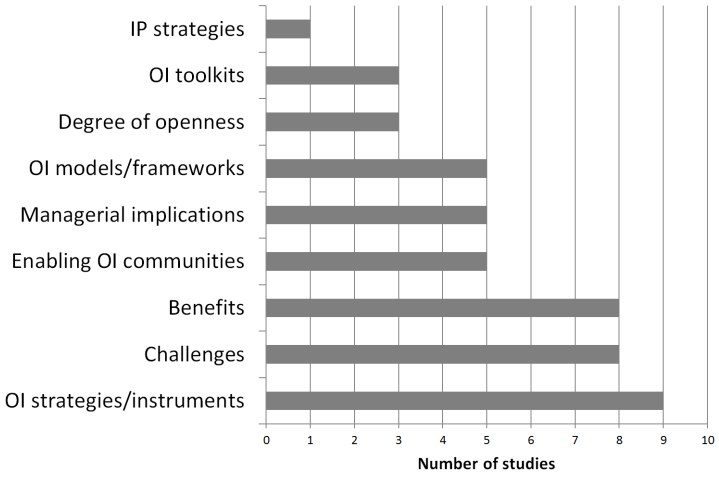
4.3 Thematic analysis

The main objective behind conducting this analysis is to find the recurring themes in the identified primary studies. Based on the guidelines provided by Cruzes et al. [30, 31], we performed the following analysis steps:

1. Extract data from the primary studies
2. Identify the interesting themes from the data
3. Group the themes into the distinct categories
4. Assess the trustworthiness of the identified themes using rigor and relevance criteria

The resulting 9 themes of OI in software engineering are depicted in Figure 4. Figure 5 provides a more detailed view on the identified themes using the mind map technique, where the 33 primary studies are referred to as **S_1** to **S_33**. In order to assess the trustworthiness of the identified themes, the rigor and relevance

Figure 4: Identified Open Innovation themes in Software Engineering



analysis is performed and its results are visualized in Figure 6. Details on the primary studies and the rigor an relevance scores are reported in the appendix, Table 1. The rigor and relevance scores are used to find the evidence in favor and against OI in SE (research question RQ2). The results from less relevant and less rigorous studies have weaker empirical support than those stemming from highly relevant and rigorously conducted primary studies. There can also be promising highly relevant studied that were conducted with low rigor.

Studies are organized into four quadrants (A, B, C and D) according to their rigor and relevance scores. The procedure for classification was as follows:

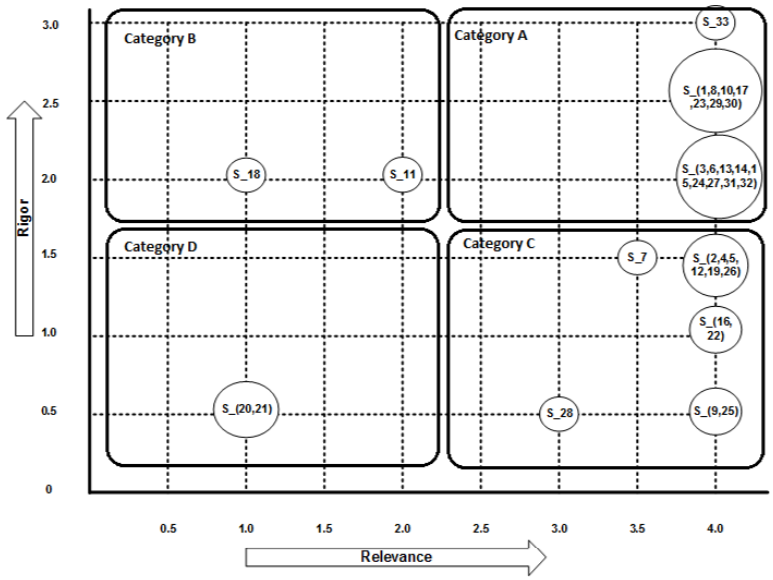
1. Studies with the score from (0–1.5) are considered as low rigor, while high rigor is defined for a score of 2 or above.

2. Studies with the score from (0–2) are considered as low relevance, while high relevance covers scores from 2.5 or above.

We classified 17 studies as having the highest rigor and relevance, see area A in Figure 6, and these results are the most trustworthy. Moreover, we classified 12 studies into C category of studies with high relevance but low rigor. On the other hand, categories B and D contain two studies each and in for both categories the relevance scores were higher than the rigor scores, see Table 1. The identified themes are are presented in the subsections below, sorted according to the number of categorized studies.

Figure 5: Mind Map of OI in SE

Figure 6: Categorization of studies based on rigor and relevance



OI Strategies/Instruments

The software industry is characterized by frequent technological changes which force large incumbent firms to more rapidly innovate their strategies in the pursuit to sustain their current revenue levels. OI strategies focus on how innovation networks and strategies can be used to participate, orchestrate or govern this technologically unstable environment.

Research and development (R&D) collaboration strategies seem to help organizations to attract and establish communities and to stay competitive. This strategy is also visible among the firms that adopt OI to enhance their innovation process in nine primary studies **(S_3, S_5, S_6, S_13, S_15, S_19, S_25, S_26, S_29)**. Six out of these studies **(S_3, S_6, S_13, S_15, S_25, S_29)** were conducted with high rigor and relevance, see category A in Figure 6. The remaining three studies **(S_5, S_19, S_26)** were classified into category C which indicate that the studies have relatively low rigor but still their results are highly relevant.

Looking at the primary studies with high rigor and relevance scores, the results of one study **(S_3)** indicated that firm's human capital affects the adoption of OI business strategy among the Finnish software companies. Consequently, the companies that have larger academically educated staff more often apply OI busi-

ness strategies. Harison and Koski **(S_3)** stated the reason for that is the ties between the OSS communities and universities. Smaller companies (start-ups) tend to apply more open innovation strategies compared to large and older firms. This interpretation seems reasonable since smaller companies often leverage OSS to acquire knowledge and substitute of a comparable depth as for the in-house R&D capabilities that they lack. Overall results suggest that a more positive attitude towards openness enables firms to better share in the benefits of open innovation processes **(S_6)**.

In a study about implementing a private collective model at Nokia **(S_13)**, a number of mitigation strategies were adopted. Nokia had the evidence of their competitors using their source code, therefore, they partially revealed their source code to retain control and information, and future plans leakage was protected through non-disclosure agreements. Moreover, the development control was compromised by involving communities, hiring key developers and upstream participation, which resulted in no single vendor being able to control the platform. Besides that, Nokia opened up and communicated the structure of its internal processes.

Dahlander and Magnusson **(S_15)** highlight that in order to address the emerging challenges of the public-private development model, such as attracting outsiders to work in their community, companies are releasing the code under open source licenses and in this way are establishing new communities or using existing communities. At the same time, companies often adopt licensing practices that clarify ownership, devoting resources to evaluate source code and give feedback on source code to communities.

One of the main conclusions of Grøtnes' study **(S_29)** is that the open innovation takes place in neutral arenas like standardization, and outside-in, inside-out and coupled processes are used to create new technological platforms. A more restricted membership gives a separate outside-in and inside-out process while open membership leads to a coupled process. A key difference can be explained by the example of Android that was available for invited firms only, while open membership is open for all. Open membership creates a modular innovation that embeds new radical innovations like mobile TV, while Android creates an architectural innovation with possibilities for further radical innovations.

Similarly, Deutsche Telekom **(S_25)** used Foresight workshops, executive forums, Customer integration, Endowed chairs (opening doors to academia world), Consortia projects (cost sharing of complex projects), Corporate Venture Capitalist (window to innovation in the start-up community and technology sourcing through co-investing), Internet platforms, Joined development, strategic alliances, spin-outs (external commercialization of internal R&D results in technologies, products or services) and test market (equipping a city with next generation infrastructure) to take advantage of open innovation, see Figure 5.

Looking at the studies performed with less rigor, West and Gallagher **(S_5)** argued that companies employing strategies such as pooled R&D/product development (firms sharing the R&D), spin-outs and selling complement and attracting

donated complements, easier overcome the following challenges: 1) the generation and contribution of external knowledge (motivating), 2) incorporating the external innovation into firms resources and capabilities (incorporating), 3)diversifying the exploitation of intellectual property (IP) resources (maximizing).

The most noted example of pooled R&D is the Mozilla project, initiated by Netscape in response the competitive pressure from Microsoft Internet Explorer (IE). Vendors such as IBM, HP and Sun needed a Unix based browser to increase sales of Internet connected workstations and therefore donated some of their IPs to the open source development lab (OSDL), while exploiting the common advantages of all the contributors to expedites the sale of related products. Similarly, spin-out (shared R&D between firms and a community) can also release the potential IP from the firm that is not creating the value anymore. Thereby, the firms transform internal development projects to externally visible open source projects.

Consequently, the donated IP generates demands for other products and services that the (donor) firms continued to sell. An examples of a spin-out is when IBM promotes the Java programming language, developed by Sun Microsystems, to compete with Microsoft. IBM was still able to generate revenue from sales of hardware and supporting services in the Java world. Selling complements is used by firms to build upon the already existing products and succeed through differentiation strategy and in contrast, donating complements are more feasible when selling to technically professional buyers, capable of making modification and improvements, such as hobbyist programmers or corporate engineers.

In addition, Dittrich and Duysters **(S_19)** also addressed the difference between exploration (seeking radical innovation) and exploitation (seeking incremental innovation) strategies adopted by firms to sustain their position in rapidly changing technological environments. Exploration networks make use of flexible legal organizational structures, whereas exploitation alliances are associated with legal structures that enable long-term collaboration. Nokia followed an exploitation (incremental innovation) strategy in the development of the first two generations of mobile telephony devices, and an exploration (radical) strategy in the development of technologies for the third generation. Such inter-firm networks seem to offer flexibility, speed, innovation, and the ability to adjust smoothly to changing market conditions and new strategic opportunities.

While studying the case of embedded Linux **(S_26)** Henkel found that hobbyists and developers in universities reveal nearly all of the code in contrast to companies. In particular, the more important it is to obtain external development support, the more code the respective firms reveal.

Challenges

This theme highlights business and process related challenges (**S_4, S_9, S_12, S_14, S_15, S_21, S_24, S_13**) faced when firms try to adopt open innovation, summarized in Table 4. Business related challenges refer to business strategy

(S_9, S_13, S_14), entry barriers **(S_15, S_21)** and governance **(S_12, S_24)**. Governance refers to establishing measurement and control mechanisms to enable project managers and software developers within the communities as well as others within a software development organization, to carry out their roles and responsibilities [25]. Process related challenges consider hinders in strategy realization.

Facets	Challenges
Business	**Business strategy** • Unclear content and contribution strategy **(S_14)** • Contribution time-line unclear **(S_14)** • Minimize modifications to the open source code **(S_14)** • Unclear relationship between the benefits from contributions in terms of strategy and business goals **(S_14)** • Be strategic when adopting innovative features **(S_14)** • Balancing the interests of those participants against those of the ecosystem leader **(S_9)** • Difficulty to differentiate **(S_13)** • Guarding business secrets **(S_13)** • Definition of core competencies **(S_21)** • Legal and property rights issues concerning the external knowledge **(S_21)** **Strategic OI entry barrier** • Accessing communities to extend the resource **(S_15)** • Reducing community entry barriers base **(S_13)** • Aligning firm strategies with the community **(S_15)** • Community build-up and management **(S_21)** • Achieving a common vision **(S_12)** • Finding staff/ Competencies **(S_24)** • Lack of expertise **(S_24)** **Governance** • Expectation management of community **(S_21)** • Increasing knowledge sharing and exchange **(S_12)** • Achieving a high level of commitment **(S_12)** • Giving up control **(S_13)** • Lack of support **(S_24)** • Lack of ownership **(S_24)**

Facets	Challenges
Process	**Agile processes** • The new approach caused significant problems in terms of transferring the ideas outside the team **(S_4)** • Visibility as to what the new [agile] team were doing dropped quickly. The introduction of agile coincided with a rapid drop in the number of developers from that team attending the overall R&D meetings. **(S_4)** • The use of short iterations, a feature backlog and stand-up meetings reduced the amount of time you can spend playing around or sharing ideas outside your team **(S_4)** • Motivating the generation and contribution of external knowledge (Motivating) **(S_4)** • Incorporating external innovation into firms resources and capabilities (Incorporating) **(S_4)** • Diversifying the exploitation of intellectual property (IP) resources (Maximizing)**(S_4)** **Relation between process and innovation** • Augmenting the requirements management process **(S_14)** • Manage innovative features in a separate process **(S_14)** • Top-down or bottom-up open innovation **(S_14)** **Release planning and prioritization** • Prioritization process needs modification **(S_14)** • Challenging acceptance criteria kills innovative features **(S_14)** • Need for special flow for innovative features to evolve to meet acceptance criteria **(S_14)** • Release planning even more challenging **(S_14)** • Prioritizing the conflicting needs of heterogeneous ecosystem participants **(S_9)** • Assimilating communities in order to integrate and share results **(S_15)** • Efficient process management **(S_21)** • Lack of Road-maps with OSS Products **(S_24)** • Overcoming Not Invented Here **(S_21)**

Table 4: OI challenges categorized in business and process themes.

As can be seen in Table 4, business and process level challenges are considered to be major hindering factors for the adoption of OI. Finding the right balance

between contributing to community and reaping benefits is tough, and thus results in unclear business strategies **(S_14)**. One of the biggest concerns is the difficulty in differentiation if a firm indulge itself in an OSS solution and guard its business secrets because its competitors have the same solution available for their products **(S_14)**. Other challenges are: managing the conflicting needs **(S_9)** of all players involved in the process, aligning the firm's strategy with community **(S_15)** and achieving a common vision **(S_12)**. Even if a firm has a clear business strategy to resolve the often conflicting stakeholders' needs, the challenge of community build up and survival remains **(S_21)**. Therefore, firms and communities need to find the right balance of governance**(S_13)**.

On the other hand, process related challenges are negatively impacting OI. For instance, Conboy and Morgan **(S_4)** suggest that agile and OI do not get along well, especially when dealing with the management of innovative requirements and release planning. Agile requirements backlogs do not have room for innovative requirements since short iterations, a feature backlog and stand up meetings make it extremely tough to play around or share ideas outside your team. The lack of control over release planning was also pointed out as a challenge in a study **(S_11)**, for example, sometimes it is a better business decision to adopt the open source code, perform minimum changes, and sell it instead of spending time on developing differentiation features. This raises a question whether or not firms should have a separate requirements management process for innovative features **(S_11)**, but nevertheless there is an inherent complexity in requirement management process while managing innovative features. Further process challenges include the lack of clear roadmaps for product highly dependent on OSS platforms and overcoming the "not invented here" mentality.

The majority of the primary studies highlighting the challenges lie in categories A **(S_13, S_14, S_15, S_24)** and C **(S_4, S_9, S_12)** suggesting that results are highly relevant to industry Only one study **(S_21)** lie in category D.

Benefits

This category highlights the OI adoption benefits in terms of positive impacts associated with the inside-out, outside-in, coupled processes and the private collective model **(S_10, S_12, S_13, S_20, S_23, S_24, S_26, S_31)**. The benefits are summarized in Table 5. As far as the strength of evidence is concerned, five papers **(S_10, S_13, S_23, S_24, S_31)** lie in category A and two studies **(S_12, S_26)** fall into category C. The fact that only one study **(S_20)** has low rigor and relevance suggests that the identified OI adaption benefits are highly relevant for industry.

Facets	Benefits
Process	**Knowledge building and exchange** • Knowledge sharing and exchange **(S_12)** • Low knowledge protection costs **(S_13)** • Easy access to all information **(S_12)** • Increases organizational learning **(S_12)** • Improves collaboration with groups in Europe, USA, India **(S_12)** • Customer demand for source code has a significant (5%), positive effect on the decision to reveal at all **(S_31)** **Platform and reuse** • Improves platform use **(S_12)** • Promotes software reuse **(S_12)** • Increases trust in platform **(S_12)** **Communication** • Direct communications **(S_12)** • Supporting OI in an existing social network site lowers the hurdles for expressing and communicating ideas **(S_20)** **Involvement and innovation support** • Improves involvement of product teams **(S_12)** • Improves feedback by being open **(S_12)** • Avoidance of duplicate work **(S_12)** • Empowers developers and project leaders **(S_12)** • Introduces diverse people to each other, adding more heterogeneous viewpoints to ideas **(S_20)** • The process acts as a catalyst for ideas: while it does not help with the initial conception of an idea, it makes all following steps easier **(S_20)** • Executing the OI might result in the realization of ideas and broadening companies offering **(S_20)** • Developer/Tester Base **(S_24)** • Flexibility of use **(S_24)**

Facets	Benefits
Business	**Time to market, cost, maintenance and efficiency** • Reduces time to market **(S_12)** • Cost savings **(S_12)** • Increases efficiency in development **(S_12)** • Reduced maintenance effort **(S_26)** • Bug fixes by others **(S_26)** • Small firms reveal significantly more due to resource scarcity **(S_26)** • Further development by others **(S_26)** **Innovation** • Increases innovative capacity and speed **(S_12)** • Adoption of innovation **(S_13)** • Increased innovation at lower costs **(S_13)** • Encourages innovation **(S_24)** • The OI technology scouting is positively associated to the SME's innovative performance **(S_10)** • Communities provide SME's a rich of free-of-charge **(S_23)** • Increases collaboration **(S_24)** **Improved competitiveness and other business gains** • Extra business functionality **(S_24)** • Improves adoption rate of the platform **(S_12)** • New competitive weapon for managers in non market leaders firms **(S_31)** • Reputation gain **(S_13)** • Revealing good code improves our company technical reputation **(S_26)** • Distribute ownership and control **(S_12)** • Learning effects **(S_13)** • De-facto standards **(S_24)** **Culture change** • Public success stories might create a culture of innovation **(S_20)** • Firm reveals all of its drivers is positively related to the importance of technical benefits **(S_31)** • External factors are less, and firm characteristics more important for selective revealing. **(S_31)**

Facets	Benefits

Table 5: OI Benefits categorized in business and process themes

The benefits are divided into the process and business related, see Table 5. OI allows firms to find a pool of skilled labor outside their boundaries without a significant cost. This external labor provides feedback and enables knowledge exchange between the community and the firms **(S_12)**. Organizational learning is another important benefit, where OI often gathers diverse people with similar interests, adding more heterogeneous viewpoints to ideas **(S_12)**. However, it is to be noticed that OI does not help with initial conception of an idea; rather it acts as a catalyst for ideas, and might also result in the idea realization. Consequently, OI provides opportunities to offer more choices to consumers and possibly broaden the firms' offerings. Furthermore, knowledge sharing and exchange lead to avoidance of duplicate work and encourages software reuse. Analyzing behavior of firms unveil that one third of the firms reveal no source code at all, and another one third of the firms reveal an amount between 0 to 100 %, while the remaining firms reveal all their source code. Customer demands are reported as the key factor that causes the firms to reveal the source code **(S_31)**.

OI also brings business advantages, outlined in Table 5. OI involvement enables efficient development processes **(S_12)**, reduces development cost **(S_12)**, and increases innovation capacity **(S_12)**. OI can also help to reduce time to market and can permit firms to build and maintain a good repute from code revealing **(S_13)**, public success stories and innovation culture **(S_20)**. Finally, findings suggest that by being open, companies can significantly increase their competitive advantage and managers from the companies that are not market leaders may consider it as the competitive weapon against their competitors **(S_12)**.

Enabling OI communities

This theme refers to communities as distributed groups of individuals, aiming at solving a general problem and/or developing a new solution supported by computer mediated communication. The solutions developed in the community can be used in conjunction with the firms' internal capability to develop competitive services and products. In particular, this theme uncovers strategies adopted by firms to use communities as complementary assets, positive impacts of the community on firms' innovation and challenges associated with it (**S_1, S_6, S_8, S_21, S_33**), see summary in Table 6.

As can be seen in Table 6, firms exploiting communities in their innovation process not only gain a good reputation but also influence the direction of development and legitimate the use of projects **(S_1, S_6)**. Having an employee in the community seems to be the key to enabler of these advantages. Thus, companies

Ref	Positive Impacts	Negative findings	Strategies
S_1	• Creates good reputation • Legitimizes the use of the project • Companies can influence the development direction for these communities	• No clear evidence that firm sponsored individuals are able to orchestrate or stimulate debate within these communities • Individuals with affiliations with large incumbents in the software industry have no significant effect in the community • Other software companies may not devote their best employees to working in the community or may only passively screen developments	• A man on the inside to be able to gain access to communities

Ref	Positive Impacts	Negative findings	Strategies
S_6	• A more positive attitude towards revealing will enable firms to better share in the benefits of open innovation processes	• Too open behavior by firms programmers would be commercially harmful • Management is not always informed about this sharing and has broad, but nonetheless limited means of monitoring it • Management may overestimate the risk of critical code leaking out	• Sponsor provides monetary rewards to contributors • Employee referrals to attract contributors • The focal firm might consider launching its own public OSS project in order to attract pragmatic OSS developers
S_8	• Feature gifts (new features instead of extension of existing features)	• N/A	• Participants having an activity (i.e. report bugs, offer bugs fix etc.) are more likely to be granted access to the developer community

Ref	Positive Impacts	Negative findings	Strategies
S_21	• Increase of ideators and therefore ideas • Strong customer orientation since users can articulate wishes directly • Possibility to use wisdom of the crowds to handle high number of ideas • New forms of evaluation with better results • More and better ideas, concepts and products • Increase in efficiency and effectivity	• Community build-up and management • Overcoming Not Invented Here • Technical realization of external interfaces • Legal and property rights issues concerning the external knowledge • Expectation management of community • Definition of core competencies • Efficient process management	• N/A

Ref	Positive Impacts	Negative findings	Strategies
S_33	• OI communities produce complementary assets that are of significant value to firms • Provide firms with the tacit knowledge to address some of the tensions that arise in firm-community interaction • Limited resource base of small firms exerts a *ceiling-effect* on the optimal level of community involvement • Above-average levels of technical community participation limit the financial performance of small OSS firms • for small firms, initial increases in involvement in the communities has a positive impact on their financial performance	• Contributing entails significant costs in terms of resource Investments and loss of strategic assets that may result in decreasing returns	• internal and external sources of innovation are complements rather than substitutes

Table 6: Studies in the OI community theme

use employee referrals or offer individuals monetary rewards to exploit communities. Besides, initiating OSS projects is an alternative way for attracting pragmatic OSS developers **(S_6)**.

Using the wisdom of crowds, and direct articulation of user wishes (**S_8, S_21**) help organizations receive new features from communities instead of extensions of already existing features. Albeit claimed that OI can bring benefits to both small and large companies, small firms with limited resources exert a ceiling effect on community involvement. OI should, in those cases, be used as a complementary asset to accelerate internal innovation and R&D processes of the organization (**S_33**). On the other hand, OI does have its cost when companies might procure the outcome of the community participation, but at the same time not be willing to devote their best resources to work in the communities (**S_1, S_6**). In addition, it remains unclear how to orchestrate or stimulate debates within communities, thereby making it hard for firms to achieve their goals. Consequently, too open behavior might be potentially harmful and contributions without selective revealing strategy could entail significant cost in terms of programming resources and loss of strategic assets that may result in decreasing returns (**S_21, S_33**). As far as the trustworthiness of results of these studies is concerned based on rigor and relevance (see Figure 6), 5 studies (**S_1, S_6, S_8, S_33**) lie in category A except for one study (**S_21**) in category D.

Managerial implications

This category includes six studies that focus on the recommendations for managers how and when to indulge in open innovation, in order to increase firms innovative performance (**S_7, S_15, S_17, S_22, S_32, S_33**), see Figure 4. The primary study **(S_17)** suggests that firms working in open source settings can pursue differentiation strategies to achieve openness, without really distancing their developers from the communities. The openness is most realized at the component level and differs significantly between software and hardware components. Openness of software seems to be more important to the community than openness of hardware. Thus, companies may get involved in open source software initiatives and secure their competitive position by capturing more value or differentiation in hardware. Managers could enhance the degree of innovation and performance of their firms in a number of ways. Among them, firms should consider getting access to skilled resources, and learning by encouraging their employees to participate in the communities, instead of free riding **(S_33)**.

Moreover, firms operating in hostile environments, motivate managers to draw knowledge out of end users and communities **(S_32)**. However, most often it is not a straightforward decision for managers to participate in communities or draw knowledge from end users. A survey conducted in Dutch software industry revealed that managers are confronted with too little available time, resources, lack of commitment, and often the wrong strategy to indulge themselves in the commu-

nities **(S_22)**. Furthermore, firms need to develop sufficient absorptive capacity to benefit from external knowledge and to find interesting tasks for community participants to keep them motivated **(S_15)**. To underline the strengths of the evidence, Figure 6 depicts four studies (**S_17, S_33, S_32, S_15**) classified in category A with high rigor and relevance, while two studies (**S_7, S_22**) fall in category C that have high industry relevance but relatively low rigor.

OI Models/Framework

This theme includes the models or frameworks **(S_7, S_11, S_13, S_20, S_27)**. Two studies **(S_27, S_13)** lie in category A and three studies **(S_7, S_11, S_20)** fall into category C, B and D respectively, see Figure 6. Jansen et al. presents an open software enterprise model (OSE) for determining the openness of a software producing organization **(S_27)**. An organization can choose to be open on both supply and demand sides of the supply chain. This happens typically by opening up development on the side of software developers and contributors, or opening up service delivery on the side of service partners who deploy, configure, and service the software platform produced by the organization. However, the paper lacks clear guidelines how to execute these activities using software engineering techniques or processes.

Stuermer et al. **(S_13)** focuses on the private-collective innovation model which proposes incentives for individuals and firms to privately invest resources to create public goods innovations. Such innovations are characterized by non-exclusivity and non-rivalry in consumption. Stuermer et al. examined Nokia's Internet Tablet development and identified five hidden costs: difficulty to differentiate, guarding business secrets, reducing community entry barriers, giving up control, and organizational inertia.

Ebner et al. **(S_7)** highlight the idea of competition as a method to nurture a virtual community for innovations. Similarly, Wnuk et al. **(S_11)** proposed a software engineering framework, designed to foster open innovation by designing and tailoring appropriate software engineering methods and tools. The framework is divided into the technical (e.g. requirements engineering, software design, development and testing techniques etc.) and methodological dimensions. Singer et al. **(S_20)** envisions a 7 step innovation process as a conceptual solution. The process covers an idea life-cycle from its creation to its realization and is exemplified on an IT-related example.

Degree of openness

Openness of a software producing organization is explicated by revealing the proprietary information. Existing and potential intellectual property rights are voluntarily given up to the interested parties in order to make them accessible. This theme comprises three studies that not only contain different forms of open strategies (**S_16, S_17**), but also presents an open enterprise software (OSE) model

developed in order to assess the openness of organizations (**S_27**). West (**S_16**) claims that proprietary platforms are more suitable for market leaders and open standards are more feasible when propriety strategies fail. Besides, differentiation can be achieved through opening some parts, by disclosing technology under such conditions that it will only provide value to customers, without really giving away the advantage to competitors. Open source provides direct benefits to many users who lack the requisite technical skills to do their own development.

Balka et al. (**S_17**) state that transparency, accessibility and replicability are important to open design communities. They present an open software enterprise model that suggests that openness can quickly create critical mass of developers or partners around the software product if the surrounding partners are prepared to enter in the ecosystem in any of roles, such as developers, values added resellers, service partners or customers. However, Balka et al. also suggest that openness is not always beneficial to the organization and mention the role of partnerships in software producing organizations as a form of openness (**S_17**). It is also noticed that openness often leads to creation of new business models. All above mentioned results carry more industry relevance, since two studies (**S_17, S_27**) lie in category A, and one study (**S_16**) in category C with high relevance and low rigor.

Intellectual property (IP) strategies

This theme refers to strategies used by firms to share IP among stakeholders in the OI context. Rayna and Striukova (**S_18**) investigated open source vs. patent pools as innovation structures, however the study has low relevance according to Figure 6. Patent pools are comprised of multi-party ownership and include not only current patents, but may also include future changes to these patents. Typically, all patents in a patent pool are available to each member of the pool. In contrast, the open source structure is based on the copy-left paradigm instead of intellectual monopoly rent paradigm, where the source code as well as any subsequent modifications and improvements are released, not only to the members of the project, but to the whole community. This study (**S_18**) compares two OI structures in terms of risks, cooperation, financial/non-financial benefits, standards and their feasibility.

Rayna and Striukova (**S_18**) argue that patent pools and open source have common risks and benefits. For instance, the key risks are associated with intellectual property right (IPR) infringement, bad publicity and discouraged further investment. On the other hand, benefits can be reaped in terms of decreased R&D expenditure and transaction cost, access to skilled resources and increased future business opportunities and reputation. Besides that, open source is exclusive in application but universal in access while patent pools are universal in application but exclusive in access. Therefore, it is more suitable for large companies to initiate or adopt patent pools compared to small companies or start-ups. Small companies may find additional benefits in terms of having a chance to set a standard

in open source and give them access to highly skilled work force, and thereby reduce the development cost. Finally, patent pools are often formed based on prior knowledge unlike open source that generates new knowledge based on skills and competences.

OI toolkits

This theme includes the toolkits developed in order to involve end users into firms' internal innovation process. Given that international firms often operate in hostile environments, limited evidence (**S_2, S_28, S_32**) was found related to the use of user innovation toolkits and its impacts on firms innovative performance. As far as the strength of evidence is concerned, one study **(S_32)** was found in category A and remaining two studies **(S_2, S_28)** fall into category C, see Figure 6. Wang et al. concluded that innovation toolkits improve the innovation outcome and productivity for users with knowledge and experience. We identified only one toolkit, namely *INOVEX* (**S_28**), that is used by software producing organizations to extract knowledge from end users. When it comes to utilizing the end user knowledge, evidence suggests that larger firms seem to exploit end users online less than the smaller firms (**S_32**). This could be due to the fact that small firms are having more open search strategies caused by a lack of skilled resources and the need to reduce the development cost.

5 Discussion

The synthesized evidence in this study suggests that smaller companies (start-ups) have higher tendency to adopt OI compared to incumbents. This trend makes sense when we consider start-ups engaging themselves in OSS solution in order to quickly acquire knowledge and R&D capabilities. OI provides initial financial gains for small companies, but also limits their financial performance when the level of participation increases above the average. Thus, in order to reap the financial benefits, it seems important to have high absorptive capacity to properly catch the technical know-how from the available knowledge. Large companies should encourage their developers to participate in communities for improved knowledge sharing and for obtaining heterogeneous viewpoints on ideas.

The primary studies classified in the OI strategies category (Section 4.3) indicate that both small and large companies explore the OI potential, but in different ways. For smaller companies, adapting OSS solutions seems to provide the most benefits, while larger companies also benefit from adapting their code ownership strategies and internally adapt OSS practices via so called inner-sourcing [66]. Therefore, it is possible to hypothesize that larger companies should dedicate more effort into the OI strategies and options analysis. Moreover, companies that own implemented assets have more possibilities to capitalize their innovative potential via OI strategies, than companies that have no implemented assets. Still, for

companies owning only intangible innovations, there exist strategies to share these assets via, for example, pooled IPR forums.

The primary studies summarized in the enabling communities for open innovation category (Section 4.3) lead to an interpretation that communities offer significant benefits that companies should exploit. In particular, it seems that initiating OSS projects is equally important from the OI perspective as joining or governing an OSS project or the entire ecosystem. It remains an important aspect to further explore what strategy is optimal, given the company's size, domain and product characteristics. Furthermore, our results suggest that firms are able to influence the direction of development (governance) in communities to some extent, with one exception. Companies that sponsor individuals in their involvement in OSS projects were not able to effectively stimulate or orchestrate debates in these projects, mainly because communities believe that companies have their vested interests in participation. This could explain the difference between OSS and OI well, where in OI organizations decide to open up when they see a potential benefit in opening up, while in OSS the community contributes with the mind set of *free software ideology* without expecting any benefits in return.

The results regarding the interplay between OI and agile methods provide interesting interpretations. It seems that openness is often compromised due to lack of transparency between competitors, and even business units within an organization. Combining agile and OI seems to create barriers in transferring the ideas outside the team's boundaries, primarily due to the use of short iterations, minimum documentation, stand-up meeting, and a feature backlog that reduces the amount of time *you can spent trying new things or sharing ideas outside your team*. The resulting lack of overall R&D group overview disables the innovation opportunities when using agile practices (**S_4**). Further, the introduction of agile with OI caused a rapid decline in teams attending R&D meetings, due to the lack of tolerance for prolonged meetings as stated by senior anonymous developers, *using the old plan-driven approach we would have been going to meeting after meeting, but since going to agile, every minute you spend in one of these meetings you just think about all of the work not being done* **(S_4)**.

To further demonstrate the challenges of OI in the agile context, developers quoted that *on-site customer practice seem to be the most telling barrier since you feel accountable to person there at all time and its harder to justify taking a half day out to sit with folks in other projects for benefits of other customers*. At the same time, managers experienced lower quality of ideas due to focus on daily work.

Managers can enhance the financial situation and innovativeness of their firms, by encouraging their employees to participate in communities. To gain further advantage, managers can consider the learning and resource advantages attached to community participation, instead of just free riding. The identified evidence suggests that participation is more strongly related to the performance of those firms that exhibit high level of social participation. However, the literature also

underlines the inherent complexity for organization to initiate, build and nurture an external community as a complementary asset to their internal R&D process. To be more specific, managers have too few resources available in order to indulge them in communities. This may lead to too much time and commitment to make significant contributions in these communities.

Business strategies also play an important role in embracing open innovation, thus companies can pursue differentiation strategies with the controlled degree of openness towards communities. Nonetheless, transparency and accessibility are important factors when talking about openness of firms. Consequently, from the firms' point of view, OI does not substitute the already existing R&D process, but it complements the existing internal innovation processes.

Regarding OI models or frameworks, fostering competing ideas seems to be promising. At the same time, companies may use social networks to lower the hurdle of sharing ideas, but since the primary study (**S_20**) presents preliminary work and therefore lacks rigor and relevance, more empirical research is needed to ensure that. Similarly, the framework presented by Wnuk and Runeson (**S_11**) is preliminary and lacks specific guidelines about which SE techniques are applicable for which contexts.

When it comes to the benefits and challenges of applying a collective innovation model (**S_20**) there is a need for further studies that directly connect benefits and challenges with SE techniques, as the current evidence is incomplete and largely anecdotal. Similarly, Jansen et al. (**S_27**) describe in great detail *what* to do rather than *how* to do it, especially on the operational level, where appropriate SE techniques can provide great support. To summarize, there seems to be a lot of interesting techniques or processes that foster OI, but the ways how to operationalize them remain unspecified and requires further research.

When looking at the results in the IP strategies theme, it appears that patent pools is an alternative solution for the companies that may not necessary have innovation implemented in software (**S_18**). Both patent pools, and OSS share many benefits and challenges, but differ in that OSS provide universal access but is exclusive in application, while patent pools restrict the access but enable application. Thus, large companies should use their IPR capital for enabling OI via patent pools.

The results indicate little research focus on the OI toolkits since only one toolkit was found among the primary studies. Moreover, primary studies suggest that extensive experience is required to unlock the full potential of these toolkits. Therefore, it remains to be explored how to enable less experienced practitioners to be more innovative and in this way to leverage their innovative potential. We believe that enabling newcomers is important to fully benefit from OI, since many OI contexts are characterized with high turnover for contributors that often contribute once in a project.

6 Implications for Research and Practice

6.1 Research Agenda for Open Innovation in Software Engineering

In line with the advice by Kitchenham et al. [78], we use the systematic mapping study to derive an agenda for further research. We interpret the increased scholarly interest in OI since the launch of Chesbrough's book in 2003 as a sign of increased importance of OI. Still the number of publications that focus on OI in SE remain small and therefore we believe that focusing on OI in SE should be highlighted on the research agenda in SE. In particular, based on the results our interpretation, the following areas should be put on the research agenda:

- Further exploring suitable software development methodologies that foster OI. The results outlined in Section 4.3 suggest that combining agile and OI provides additional challenges that may have a ceiling effect on the potential benefits from OI. Thus, it is important to direct research efforts into better understanding of which development methods or processes best suits OI and what changes need to be implemented to unlock OI's full potential.

- Providing clear managerial guidelines on how to adapt OI depending on the context factors, with a special focus on which SE techniques, processes and methods can be applied depending on the selected managerial strategy. In this way the findings reported in Sections 4.3 and 4.3 will be complemented by guidelines on the operational level to form more complete solutions for adapting to OI.

- Exploring the balance between community involvement and in-house SE activities. This study identified several benefits from OI and OSS community involvement, see Section 4.3. However, the process-related benefits should be further explored, with a focus on uncovering where involvement brings most benefits. In particular, the role of OI involvement in improved testing remains unexplored, where we believe that OI provides not only significant reduction of the test effort but also can be a source of innovation. We base this assumption on a premise that testing uncovers unexpected behavior of software, which could be inspirational in the innovation process.

- Focusing on the role of requirements engineering in OI both during and beyond innovation discovery. OI offers access to a wide and heterogeneous communities of potential stakeholders which puts pressure on the current techniques for key stakeholder identification and domain understanding. Advances in current techniques are required for supporting the identification of commodity and competitive advantage requirements sources. Beyond innovation discovery, there is a need for a decision making support that can

combine both strategic and operational levels and provide run-time requirements triage support for capturing and incorporating OI potential into product planning and requirements decision making. Despite that, researching if unimplemented requirements that represent valuable IPRs, can be shared with others in a similar way as for example patents, and what benefits this approach brings is important.

- We encourage researchers to develop and publish more solution and validation research in OI as these remain underrepresented, see Figure 3. The large number of evaluation research is definitely positive but, at the same time, highlights the immaturity of the OI in SE research area. Thus, more solutions in terms of tool proposals or frameworks and their validations are needed to advance to the next maturity level.

6.2 Implications for industry practice

Although we summarize the empirical evidence in the field of OI in SE being scarce, there is some evidence that may be used to guide software companies in their innovation strategies:

- The identified conflict between agile and OI principles should be given special attention. Agile principles focus developers attention and communication in order to meet specific project goals. However, the innovation process benefits from the noise of leaks in the information flow from multiple sources, internal as well as external. Companies should make sure that this information flow is regained, using other practices.

- Open innovation strategies seem to be more beneficial for smaller and newer actors in a market. They may apply OI and thus can gain significant competitive advantage against competitors by more quickly absorbing potential innovation. However, there are also examples of major corporations that manage a software ecosystem, based on open or semi-open innovation. The take-away for companies is that they need to define and monitor their OI strategy to make sure their actions are relevant, given their current and future expected market position.

- An implication for industry, based on the literature findings, is that OSS and OI is *not* for free. In order to gain the full and long term benefits from OI, companies must invest in the open communities, and since these are complex networks with a multitude of actors, these companies must have a clear resources investment plan, just as they need for closed innovations.

- IPR management is different for OI. The studied research recommend large companies using patent pools to manage their IPR capital in relation to the open innovation community.

7 Conclusions

Open innovation (OI) becomes significantly important for companies developing software-intensive products and services. It provides several benefits that force these companies to re-think and often significantly change their current innovation strategies. The external availability of innovations combined with the flexibility of their realization generate new opportunities for providing value to the customers. OI pushes software industry into a new ground where well-known and checked software development and management strategies need to be revisited. At the same time, OI remains greatly unexplored in the SE literature, focusing greatly on exploring OSS, resulting in a lack of systematic efforts to summarize OI literature in relation to software engineering.

We conducted a systematic mapping study on OI in software engineering with the aim to identify the existing themes in the literature and evaluate them based on the rigor and relevance analysis.

Answering research question *RQ1* we identified nine themes. The dominant themes are related to OI strategies, OI challenges and benefits, enabling OI communities, managerial implications of adaption OI and OI models or frameworks. The degree of openness, OI toolkits and IP strategies are less frequently represented in the surveyed papers.

Our findings for *RQ2* suggest that the majority of the studies is conducted with high rigor and high relevance (17 out of 33) and as many as 29 out of 33 were considered industry relevant. This strongly indicates that OI in SE is industry practice oriented. Further, 27/33 studies are of evaluation type, which is unusually high for a mapping topic. Therefore, we encourage more solution and validation research, see Table 1. The high rigor and relevance scores also imply generalizability of the results derived from thematic analysis in answer to *RQ1*.

This mapping study leads to a proposal to further explore SE in OI in terms of development methodologies that interplay with OI, situated managerial guidelines for OI adaptation, as well as exploring the balance between open community involvement and in-house development. Specifically, the roles of testing and requirements engineering in OI remain unexplored.

Acknowledgement

This work is funded by the Swedish National Science Foundation Framework Grant for Strategic Research in Information and Communication Technology, project Synergies (Synthesis of a Software Engineering Framework for Open Innovation through Empirical Research), grant 621-2012-5354.

APPENDIX A

RIGOR AND RELEVANCE CRITERIA

1 Rigor

Context(C)

1. **Strong description:** The context is described to the extent where it becomes comparable to other settings [71]. In particular, we emphasized subject type (graduate, undergraduate, professionals, researcher), development experience, development methodology, duration of the observation. If all these aforementioned factors are highlighted, then C is evaluated to 1.

2. **Medium description:** If any of the above mentioned factors is missing in the study, then C is evaluated to 0.5.

3. **Weak description:** If no description of context is provided in the study, then C is evaluated to 0.

Design (D)

1. **Strong description:** The research design is described to the extent where it becomes transparent and detailed enough for the reader to understand the design [71]. To be specific, if the study underlined the outcome variables, measurement criteria, treatments, number of subjects , and sampling, then D is evaluated to 1.

2. **Medium description:** If a study is missing out on any of the factors related to design and data collection is missing (see above), then D evaluates to 0.5.

3. **Weak description:** If no design description is provided at all then, D is evaluated to 0.

Validity threats (V)

1. **Strong description:** If different types of validity (i.e. internal, external, conclusion and construct validity) are evaluated and reflected upon then, V is evaluated to 1.
2. **Medium description:** If a study only highlights the subset of the relevant threat categories then, V is evaluated to 0.5
3. **Weak description:** If a study is missing out on validity discussion completely, then V is evaluated to 0.

2 Relevance

Users/Subjects (U)

1. **Contribute to relevance:** If the subjects used in the study are from industry (professionals) then, U is evaluated to 1 for industry.
2. **Partially contribute to relevance:** The subjects are partially representative, i.e. they are master(Msc.) or graduated students then, U is evaluated to 0.5
3. **Does not contribute to relevance:** If the subjects are bachelor/undergrad students or the information is missing then, U is evaluated to 0

Scale (S)

1. **Contribute to relevance:** If an industrial size application is used in the study then, S is evaluated to 1.
2. **Does not contribute to relevance:** The application is down-scaled or a toy example hence, S is evaluated to 0.

Research Methodology (RM)

1. **Contribute to relevance:** The chosen research methodology is suitable to scrutinize real world contexts and situations with relevance for practitioners (action research, case study, industry interviews, experiment investigating a real situation, and surveys/interviews). If study belongs to any of the aforementioned research methodologies then, RM is evaluated to 1
2. **Does not contribute to relevance:** If a Study is using Lab experiment (human subjects/software) or missing information then, RM is evaluated to 0.

Context (C)

1. **Contribute to relevance:** If a study is executed in a setting that matches real industrial usage (industrial setting) then, C is evaluated to 1.
2. **Does not contribute to relevance:** If a study is investigated under under artificial setting (e.g. lab) or others that do not represent a context matching real world situations, or not reported then, C is evaluated to 0.

Table 1: Rigor and relevance scores with category

Study_ID	Ref.	C	D	V	Rig. Sum	U	S	RM	C	Rel. SUM	Category
S_1	**[34]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_2	**[137]**	0.5	0.5	0.5	**1.5**	1	1	1	1	**4**	C
S_3	**[57]**	1	1	0	**2**	1	1	1	1	**4**	A
S_4	**[28]**	1	0.5	0	**1.5**	1	1	1	1	**4**	C
S_5	**[139]**	1	0.5	0	**1.5**	1	1	1	1	**4**	C
S_6	**[63]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_7	**[39]**	0.5	0.5	0.5	**1.5**	0.5	1	1	1	**3.5**	C
S_8	**[136]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_9	**[144]**	0.5	0	0	**0.5**	1	1	1	1	**4**	C
S_10	**[110]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_11	**[150]**	1	0.5	0.5	**2**	0	0	1	1	**2**	B
S_12	**[101]**	0.5	0.5	0.5	**1.5**	1	1	1	1	**4**	C
S_13	**[132]**	1	1	0	**2**	1	1	1	1	**4**	A
S_14	**[148]**	0.5	0.5	1	**2**	1	1	1	1	**4**	A
S_15	**[33]**	1	1	0	**2**	1	1	1	1	**4**	A
S_16	**[140]**	0.5	0.5	0	**1**	1	1	1	1	**4**	C
S_17	**[9]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_18	**[117]**	0.5	1	0.5	**2**	0	1	0	0	**1**	B
S_19	**[37]**	0.5	0.5	0.5	**1.5**	1	1	1	1	**4**	C
S_20	**[127]**	0	0.5	0	**0.5**	0	0	1	0	**1**	D
S_21	**[70]**	0	0.5	0	**0.5**	0	0	0	1	**1**	D
S_22	**[133]**	0.5	0.5	0	**1**	1	1	1	1	**4**	C
S_23	**[26]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_24	**[102]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_25	**[120]**	0.5	0	0	**0.5**	1	1	1	1	**4**	C
S_26	**[62]**	1	0.5	0	**1.5**	1	1	1	1	**4**	C
S_27	**[73]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_28	**[18]**	0	0.5	0	**0.5**	1	1	0	1	**3**	C
S_29	**[55]**	1	0.5	1	**2.5**	1	1	1	1	**4**	A
S_30	**[36]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_31	**[64]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_32	**[85]**	1	1	0	**2**	1	1	1	1	**4**	A
S_33	**[131]**	1	1	1	**3**	1	1	1	1	**4**	A

APPENDIX B

DATABASE SEARCH STRINGS

Search string used for the Compendex and Inspect database (years 1969 to 2013):

(((((Open Innovation WN KY OR Open-Innovation OR OI WN KY OR innovation WN KY OR innovation management WN KY) AND (software WN KY OR software ecosystem WN KY OR product line WN KY OR requirement* engineer* WN KY OR requirement* management WN KY OR open source WN KY) AND (exploratory study WN KY OR lesson* learn* WN KY OR challenge* WN KY OR guideline* WN KY OR Empirical investigation WN KY OR case study WN KY OR survey WN KY OR literature study WN KY OR literature review WN KY OR interview* WN KY OR experiment* WN KY OR questionnaire WN KY OR observation* WN KY OR quantitative study WN KY OR factor* WN KY) AND (ENGLISH) WN LA))))

Search string used for the ACM Digital Library database (years 1969 to 2013):

((((((((((((((((("Title":"Open Innovation" OR "Title":"Open-innovation" OR "Title":OI OR "Title":innovation OR "Title":innovation management) AND ("Abstract": software OR "Abstract": software ecosystem OR "Abstract": requirement* engineer* OR "Abstract": open source OR "Abstract": product line) AND ("Abstract":exploratory study OR "Abstract": challenge* OR "Abstract": guideline* OR "Abstract": Empirical investigation OR "Abstract": case study OR "Abstract": survey OR "Abstract": literature study OR "Abstract": literature review OR "Abstract": interview* OR "Abstract": experiment* OR "Abstract": questionnaire OR "Abstract":observation* OR "Abstract":quantitative study OR "Abstract":factor*))))) and (FtFlag:yes))) and (FtFlag:yes)) and (PublishedAs:journal OR PublishedAs:proceeding OR PublishedAs:transaction) and (FtFlag:yes)))) and (PublishedAs:journal OR PublishedAs:proceeding OR PublishedAs:transaction) and (FtFlag:yes)))))))))

Search string used for the IEEE Explore database (years 1969 to 2013):

((("Index Terms":"Open Innovation" OR "Index Terms": "Open-Innovation" OR "Index Terms":OI OR "Index Terms": innovation management OR "Index Terms": innovation) AND (Search_Index_Terms: software OR "Index Terms":

ecosystem OR "Index Terms": product line OR "Index Terms":requirement* engineer* OR "Index Terms": requirement* management* OR "Index Terms": open source) AND (p_Abstract:case study OR "Abstract": exploratory study OR "Abstract": lessons learn* OR "Abstract": survey OR "Abstract": Empirical investigation OR "Abstract": guidelines "Abstract": literature study OR "Abstract": interview OR "Abstract": experiment OR "Abstract": factors OR "Abstract": questionnaire)))

Search string used for the ISI Web of Science database (years 1969 to 2013):

(((TI=("Open Innovation" OR "Open-Innovation" OR OI OR innovation OR innovation management) AND TS=(software OR software ecosystem OR product line OR requirement* engineer* OR requirement* management OR open source) AND TS=(exploratory study OR lesson* learn* OR challenge* OR guideline* OR Empirical investigation OR case study OR survey OR literature study OR literature review OR interview* OR experiment* OR questionnaire OR observation* OR quantitative study OR factor*)))) AND Language=(English) Refined by: Web of Science Categories=(COMPUTER SCIENCE INFORMATION SYSTEMS) Timespan=1969-2013. Databases=SCI-EXPANDED, SSCI, A&HCI, CPCI-S, CPCI-SSH

Search string used for the Science Direct database (years 1969 to 2013):

(open innovation OR open-innovation OR OI OR innovation OR innovation management) AND (software OR software ecosystem OR product line OR requirement* engineer* OR requirement* management OR open source) AND (exploratory study OR lesson* learn* OR challenge* OR guideline* OR Empirical investigation OR case study OR literature study OR literature review OR interview* OR experiment* OR case study OR questionnaire OR observation* OR quantitative study OR factor*)[All Sources(Computer Science)]

CHAPTER II

A SURVEY ON THE PERCEPTION OF INNOVATION IN A LARGE PRODUCT-FOCUSED SOFTWARE ORGANIZATION

Abstract

Context. Innovation is promoted in companies to help them stay competitive. Four types of innovation are defined: product, process, business, and organizational.
Objective. We want to understand the perception of the innovation concept in industry, and particularly how the innovation types relate to each other.
Method. We launched a survey at a branch of a multi-national corporation.
Results. From a qualitative analysis of the 229 responses, we see that the understanding of the innovation concept is somewhat narrow, and mostly related to product innovation. A majority of respondents indicate that product innovation triggers process, business, and organizational innovation, rather than vice versa. However, there is a complex inter-dependency between the types. We also identify challenges related to each of the types.
Conclusion. Increasing awareness and knowledge of different types of innovation, may improve the innovation. Further, they cannot be handled one by one, but in their interdependent relations.

1 Introduction

In recent years, the focus on innovation has increased in many lines of business. Novel products and services have always been important, while with an increas-

ing pace of change, new technologies and market concepts being launched, with small vendors coming up and changing the scene in very short time, the need for continuous innovation is stressed in larger companies. Internet technologies for communication and distribution, and products and services primarily differentiated with respect to software, enables this shift by lowering the thresholds for new actors, and thereby threatening the position of existing ones.

Innovation is not only bringing new products to the market. The Organisation for Economic Co-operation and Development (OECD) Oslo manual [4], which is used to guide national statistics collection on innovation, distinguishes between four categories of innovation, i) product, ii) process, iii) marketing, and iv) organizational. These categories are defined as follows: *A product innovation is the introduction of a good or service that is new or significantly improved with respect to its characteristics or intended uses* [4, §156], while a *process innovation is the implementation of a new or significantly improved production or delivery method* [4, §163]. In the context of software engineering, we also count software development processes and practices as “production" methods in the process innovation category. *A marketing innovation is the implementation of a new marketing method involving significant changes in product design or packaging, product placement, product promotion or pricing* [4, §169]. Note that this involves the whole concept of bringing a product or service to the market, a kind of innovation we have seen in the software and internet domain, for example, using information or advertising instead of money as a trade for services. Finally, an *organizational innovation is the implementation of a new organizational method in the firm's business practices, workplace organization or external relations* [4, §177]. This is also prevalent in software, where for example open source software, outsourcing and offshoring significantly has changed the game in many lines of business.

Given these categories of innovation, we were interested in studying to what extent these were known and integrated in the culture of a large company, which is under rapid change, and where innovation is a key survival factor, due to the volatility of the market. In particular, we wanted to study the awareness of the innovation concepts, and the interplay between the four types of innovation; which types precedes the other? There is a similarity to the software process improvement trinity of people, process and technology, much discussed in the 1990's [69]. More specifically, this study formulates three research question:

RQ1 What are the general perceptions of the term *innovation*?

RQ2 What relations are assumed between *product* innovation and *process, organizational* and *marketing* innovation, respectively?

RQ3 Which challenges exist with respect to the four types of innovation?

To address the research questions we launched an internal online survey [46] in a local branch of a multi-national corporation. The target population consisted

of approximately 900 employees. On a global level the company employs approximately 5,000.

We found that the understanding of the innovation concept is somewhat narrow, and mostly related to product innovation. A majority of respondents indicate that product innovation triggers process, business, and organizational innovation, rather than vice versa. However, there is a complex inter-dependency between the types.

The paper is outlined as follows. In Section 2 we summarize empirical studies on people's attitudes to innovation in software engineering. Section 3 describes the methodology and design of the survey, as well as threats to validity and a characterization of the case company. In Section 4, we report our findings from the survey, and analyze the data. Section 5 concludes the paper.

2 Related work

Innovation related to information technology (IT) has become vital part of most organizations' success, primarily for two reasons: i) growing importance of innovation for organizational life, and ii) the introduction of IT into almost every business unit of organizations [45]. Lee and Xia [89] addressed the process bottlenecks to innovation, where development teams are inefficient and reactive in most cases. Consequently, this causes problems with lack of support for business adaptions to shifting demands. Agile development seem to offer remedy to make the whole process more innovative for product development and help development teams to quickly deliver innovative, high quality solutions to an ever increasing demand of business innovation [65].

On the other hand, research evidence [29] also suggest that agile could also be a hindrance for product innovation. It creates barrier in transferring the ideas outside the team boundaries due to short iterations and feature backlog reduced the amount of time that teams could spent trying new things or sharing new ideas across different teams. Wnuk et al. [149] also hinted the fact that existing requirements processes are designed to handle mature features and consequently, raises the question of process innovation by having a separate requirements engineering process to make room for innovative features (other than featured backlog) in the products.

Lund at al. [99] conducted a survey to explore the effects that reutilization have on innovation. Results revealed that standardization of process will free up time for innovation and most interestingly, routines are capable of having positive impact on occurrence of ideas and follow through on ideas. Furthermore, paring routines with openness to continuously improve the existing routines leverage positive effects on innovation. Therefore, take away from the study for managers is to take a look at existing routines with the spectacle of improving them, which will not only improve the efficiency but also the innovation aspect.

Moreover, another study was found where Harrison et al. [58] conducted a survey with 170 Finnish software organizations to explore the impact of human capital on open innovation. Therefore, it can be used as an example where people are affecting the innovation activities in the organization. The study findings suggest that software companies with the larger academically educated staff are more likely to apply open innovation business strategies to accelerate their internal innovation process. The study further argued that this could be due the strong ties between communities and universities. Similarly, Nirjar [108] also performed a survey with 121 software companies across India to explore the impact of workforce commitment on the innovation capability of the software enterprises. The study findings highlighted that the commitment of the managers of software firms can significantly enhance the innovation productivity by creating certain policies (i.e. open business model) [24] and practices/processes.

3 Methodology

In this section we describe the surveyed company more thoroughly and elaborate on the survey design, analysis and threats to validity.

3.1 About the company

The company, which is a multi-national corporation with approximately 5,000 employees globally, develop embedded devices and the studied branch is focused on software development for communication hubs and additional connected devices in an internet of things (IoT) fashion. We consider the studied company a representative case [122] for similar ones, and hypothesize that the findings have a much broader generality than just this company. The studied branch of the company has 1,600 employees, of which 800 work on software development for the devices, and 100 work on connected devices.

The company develops software in an agile fashion and uses software product line management (SPL) [114]. The company has defined more than 20,000 features and system requirements across all the product lines. Considering the innovation aspect, the company is moving from a closed innovation model to an open innovation model [24], through the use of open source software to exploit the external resources to accelerate their innovation process. The open source solution, referred to as *the platform*, is the base for their software product line projects and derived products. New projects on the product line typically entails 60 to 80 new features with an average of 12 new system requirements per feature. There are more than 20 to 25 development teams develop these features.

3.2 Survey design

An internal online survey [46] was designed in collaboration between the researchers and company representatives, running an internal project, aimed at assessing and improving the innovation climate in the company. The questionnaire is composed of three major parts:

1. Factors that contribute to the innovation climate, based on Ekvall's scheme [41].
2. Questions on the four types of innovation (product, process, organizational and marketing) and their relation, based on the OECD model [4].
3. Factors that hinder and help innovation, based on Jansen et al.'s Open Software Enterprise model [73].

In addition to ranking and preference questions, the survey had fields for free input for most questions. The questions were defined in several iterations between researchers and company representatives, particularly to make the terminology of the survey understandable for the participants. Further, the survey was piloted to a small group of company representatives before the final launch.

One particular term was given certain care, namely *marketing innovation*. The original definition is that a *marketing innovation is the implementation of a new marketing method involving significant changes in product design or packaging, product placement, product promotion or pricing* [4, §169]. However, in the company context, the term was perceived to be only related to what the marketing department was responsible for, and thus too narrow. Therefore, we replaced the term with *business innovation* and extended it to cover the process where the needs of the customers are captured as input for the product planning. This extends business innovation into the area of Requirements Engineering, which can be seen as a software engineering process, i.e. is covered by the process innovation definition. This area is therefore somewhat overlapped, but with the general distinction that high level capturing of requirements is mainly covered by the business innovation definition.

The survey was launched via the company intranet in October and November 2013 to about 900 employees via a census sampling, most of them being developers, of which 229 responded, i.e. a response rate of 25%.

3.3 Survey analysis

As the surveyed company is product-focused the surveys had a main focus on determining the level and perception of product innovation. Due to the attempt to address the more general innovation questions, the analysis focuses on three of the questions, connecting product innovation to process, business and organizational innovation.

The respondents were asked to "select the more likely scenario" in the following questions:

- The product innovation triggers the process innovation, or vice versa
- The product innovation triggers the business innovation, or vice versa
- The product innovation triggers the organizational innovation, or vice versa

This gave an ordinal scale with two options to answer which makes any attempt of drawing conclusions limited, although a general pattern was observed, as shown in Figure 1. The survey generated 469 free text comments. Except for the three earlier mentioned questions, comments were mainly gathered from four questions where the respondents were asked how innovative (s)he perceived the organization to be with respect to the four types of innovation.

Qualitative analysis with a thematic approach [31] was used to analyze the data, which was codified in up to three levels. Based on the codified data and the comments in general, perception of innovation concepts were analyzed (Subsection 4.1) and the connections between product innovation and process, business and organizational innovation, respectively were identified (Subsections 4.2–4.4). Further on, based on the themes and comments in general, challenges were then identified and generalized in regards to the four types of innovations (Subsections 4.5–4.8).

3.4 Threats to validity

The construct validity [79], refers to whether the survey measured what it was intended to. This can be addressed through e.g. pilot studies, which was performed before the official launch. Further on, the questions were developed iteratively and based on established literature.

In regards to the analysis, a threat to the construct validity is the risk of researcher subjectivity as the first author performed the mapping and main analysis. This was addressed by having the second and third authors perform their own individual analysis of the data, and could compare their findings with that of the first author.

External validity regards whether the results be generalized to outside of the surveyed sample [122]. In this paper, we analyze the questions, which can be published from the company's confidentiality perspective. Thus, we do not focus on their perceived current innovation status, but rather on the general understanding of innovation factors and their relations. Thereby, we also focus on the most generalizable aspects, which we hypothesize are valid for other companies of similar characteristic to the studied one, as a representative case [122].

A surveys reliability [79] concerns whether the same results can be obtained if the survey process was repeated. As the sample was obtained through a census sampling frame and had a response rate of 25% we regard this optimistically. Although, this cannot be strengthened until follow-up surveys are performed. This is something that will be done in the future as the company wants to measure how the internal perception of innovation develops over time.

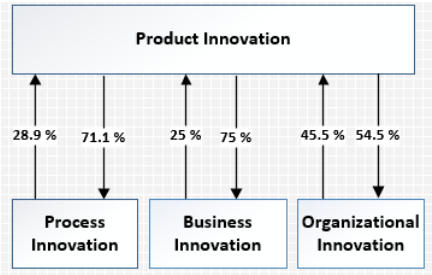


Figure 1: Triggering relation between the four types of innovation: product, process, business and organizational. Percentage value shows the share of respondents that select X → Y as the most likely scenario.

4 Results

In this section we present our findings from the qualitative analysis of the survey responses. First the general perceptions of innovation is presented based on survey responses in Fig.1. Then connections between product innovation and process, business and organizational innovation is presented respectively. Direction of arrows show the innovation type triggering the leading innovation (see fig. 1). For instance, the arrow from process innovation to product innovation shows that 28.9% respondents think that process innovation leads to product innovation. Similarly, the arrow from product innovation to process innovation suggest that 71.1% respondents think that product innovation lead to process innovation and the same arrow pattern applies for other innovation types. Finally, the challenges identified in regards to each innovation type is listed. As the types of innovation relate to each other, the challenges are structured accruing to the type where it relates the most, although a challenge may affect more

4.1 Perceptions of innovation

Although not general, it was observed among the comments that some had trouble relating to the term innovation as such. The borderline between when something goes from being an improvement or common functionality to an innovation is fluid. *"I recognize that [company] does this often [. . .] But I'm not sure if it's really innovative or just mindless changes."*

Some respondents consider innovation as part of their everyday work, while others are a bit more unclear on the distinction between their everyday work and innovative activities, or just creativity as a process. *"As a designer the largest part of the task when bringing forward is to be creative. However there is a difference between being creative and being innovative."*

A reason could be unawareness of what the company counts as innovations and examples of different types of innovations. *"I don't know much about the*

innovations that we do. I didn't know about the [example feature] for instance".

Some may not be aware of what they do could actually count as an innovative activity. *"I work with support systems and not product development. Some part of the time goes into improving how we produce products."*

Further on, some believed that they were not able to perform any innovative activities as it was not a part of their work description or role. A tester expressed how he was not able to innovate as he assumed this was a task dedicated to developers. Another tester reasoned similarly. *"Working with testing so not much improvement in the product besides some ideas that pops up occasionally."*

This thinking was present on a general level in connection to all of the four types of innovation. As mentioned, this could be due to that the awareness is limited of how and where they can innovate. A better understanding needs to be achieved for the different types of innovations and how these interplay. *"Most of all, I would say that I have only minor insight and understanding of this field [of organizational innovation]."*

A consequence may be that some believe innovation is not possible. *"I don't think it is possible to be innovative in this area [organizational innovation]."*

Apart from spreading awareness and knowledge, another important factor that needs consideration is the mindset. *"Since I'm not involved in this part of our business then it's not in my mindset, but when you now mentioned it I will take it into my consideration of innovation.".*

4.2 Product innovation vs Process innovation

On the question whether product innovation triggers process innovation, or the other way around, 71 percent answered the former (see fig.1). Although the percentage points in one direction, it is clear from the free text answers that this question is more complex than so.

Processes can be strict and complex, creating overhead and distraction, occupying time that could have been focused on creative thinking, as pointed out by a respondent. *"If the development process is driven as a rigid framework that is complex and difficult to understand who decides what and why, then you do not get in the dynamics of ideas."*

This is also identified as a challenge of process complexity in Subsection 4.6. Although processes can force a static frame on employees, it can help to bring structure to the innovation process and thereby still encourage innovation and creative thinking. *". . . well defined and established processes leads to innovative products."*

Another challenge is idea tracing and execution uncertainty (see Subsection 4.5), which is an area where we hypothesize that well-designed processes can help to clarify what happens to ideas and the roadmap for how innovations can be pushed through. Similarly, processes can also help to increase the awareness of the product scope and the innovation strategies in the organization.

Process innovation may help the organization become more efficient and reduce waste as can be interpreted by the OECD definition [4] and as pointed out by a respondent: *"... process innovation improve performance, simplifies and speeds-up development process - thus allowing to have more resources in true product innovation"*. This aligns with the area of Software Process Improvement [59], which includes possible implications from new or improved tools and techniques. As put by another respondent: *"... We need to have the proper techniques, equipment and SW in order to develop new and improved products."*

The resources made available can be defined as freed-up budget-hours, which can be used for other purposes, such as time dedicated to activities focused on rendering product innovation. An organizational and cultural challenge in this case is to actually make this dedication which demands a committed management. *"The process innovations are often meant to make development faster with more quality, but I'm not sure the gained resources are spent on product innovation."*

Beneficial factors from a process change, other than freed up resources, may also include an increase in performance and quality as confirmed by the respondents. Although, it is a matter of definition how software quality relate to product innovation [115], this will hopefully render in a better product offering which further down the release ladder may prove to be a trigger of future product innovations.

Hence, by innovating and improving the processes in the correct way and dedicating the freed up resources to product innovation, process innovation can be seen as a trigger for product innovation. This is in line with findings by Lund and Magnusson [99]. On the other hand, processes are not decoupled from the products. There needs to be an awareness of product roadmaps and an adaptive mindset as some processes may require continuous tailoring as a consequence. *"I think the general mindset is "keeping the eye on the prize", you see the upcoming releases in the horizon and you adjust the process to meet those releases."*

The need to adapt is not a simple task and requires both resources and dedication. Keeping pace with new features and products can be very demanding for an organization as pointed out by the respondents. Process changes needs to be quickly adopted for the organization not to fall behind or get confused, as described in the process innovation challenges (Subsection 4.6).

Just as new products may create a demand for new processes and tools, they can also be an inspiration for new techniques and solutions. *"On the other hand, new products can also inspire new techniques and HW/SW solutions."*.

4.3 Product innovation vs Business innovation

On the question whether product innovation triggers business innovation, or the other way around, 75 percent answered the former (see fig.1). As with the previous question, although there is a clear majority in one direction, this does not give the complete answer.

Some see product innovation as the driver with respect to business innovation due to that *"Innovative products are a great source for new business opportunities and marketing"*. Innovative features affects which consumer groups that should be targeted, and in effect which marketing channels that can be used. The nature of the innovative features also has implications on how the marketing message can be phrased and communicated. From this point of view, the products both enable and set a demand for a continuous business innovation that can adapt to changing functionality and feature sets. A good product as foundation, can even be seen as a source of inspiration to excel business innovation as hinted by the following respondent. *"I think everything starts with the product. If you are a company with "Wow!"-products then the rest will come. A consumer will see through (eventually) if the company is only selling a mediocre product but have brilliant marketing. However, if we have good products, it will be more motivating bringing it to the market, which will inspire us to excel also in business innovation"*

From the other perspective, innovative marketing may be a requirement for what otherwise would be considered a normal product. Competitive products, which are technically inferior, may very well prove more popular compared to a technically superior product, due to the awareness and visibility towards the customers, as identified by the respondents. Business innovation can create the hype needed to tell about what the innovative features are, how they differentiate and how they fit in the customers' context. However, as pointed out by the previous quote, if the product does not fill the expectations, innovative marketing will not be a viable solution in the long run.

New innovative ways are continuously needed to keep pace and capture the demands from the existing and emerging customer channels, e.g. through end-user feedback [10]. An awareness of what needs the customers have today and will have tomorrow, is an important input from business and marketing to push the product innovations forward in the right directions. *"Because business innovation brings in new experience directly from market, new demands and requirements and thus giving a product a right direction"*

This creates a challenge for the organization in terms of synchronization. The view of what features are to be considered game-changers and prioritized in the release planning process [20], may prove troublesome due to internal communication gaps between marketing and product development [75], which may lead to wrong features being promoted as a consequence. *"Scope/product planning, business side and development [should be] in sync regarding both our innovation initiative [. . .] and how to drive innovations all the way to product."*

As explained, there is a dual sided relationship. There is a dependency going in both directions where one can trigger the other. One respondent provided a concrete example which summarizes the relationship. *"It is pretty much both. Look at the music and film business which has invented new ways of marketing and distribution, but I believe the wish of distribute TV via satellite has created new products for making it possible and to get paid for it. Then again we have the*

Google glasses. Right now they are cool, but not very useful until we find a useful feature for them and that itself will create a business for them."

4.4 Product innovation vs Organizational innovation

On the question whether product innovation triggers organizational innovation, or the other way around, 55 percent answered the former (see fig.1). Opposed to the previous questions, this was not as clear majority for the product innovation centric view.

Improving and innovating the way in which a company collaborates and interacts with external parties and stakeholder, can trigger product innovations in several ways. Application of open innovation business strategies is one way to accelerate their internal innovation process [58]. Crowdsourcing ideas, engaging in Open Source communities, welcoming third-party developers, acquiring promising startups and starting joint-ventures or ecosystems are a couple of activities that falls into the open innovation paradigm originally defined by Chesbrough [24], that may render in new product innovations.

Creating a more innovative organizational environment with committed employees is another way that can lead to more product innovations [108], as described by a respondent: *"With a flexible and happy organization that makes people get looser boundaries I believe we can get a more innovative climate"* Bringing people from different backgrounds and functional areas creates diversity and enables for new discussion to arise and to discuss ideas from new angles [19, 81], or as put by the following respondent: *"Connecting colleagues which hadn't possibility to communicate before allows to discuss more problems and ideas."*. Calantone et al. [19] adds that this cross-functional integration also allows for the employees to evolve their skills by learning and sharing knowledge amongst each other, which is important for product development.

This connects to a need for a general awareness of what has been done, and what is being worked on. *"... more often than not these innovations are "hidden" in small segments of the company, not actively promoted and spread (and that's both good and bad, many projects dies when they need to become too big)."* By communicating items such as features, functionality, experienced problems and related solution across internal borders, cross-functional views can be established more automatically. A solution in one project may turn out to solve the same issue or create new ideas in another project, which could either be considered a process or a product innovation. This relates to the concept of inner source [94] and how it can help organizations work more open and cross-functional, and in the end become more innovative [**?**].

Organizational barriers and communication issues is another area, where organizational innovation may trigger product innovation in the long term perspective. When products or processes stretch over multiple business units or projects, this can create room for bureaucracy, different prioritization schemes, culture and pol-

itics, to mention a few factors [81]. *"Some sections within the company are quite innovative, but when it comes to cross-functional agreements and alignment, there always seems to be a resistance to change and adapt to new ways of working and safeguarding what seems to the best for "me/my team" is more important than what's best for the company."*

Pushing through and spreading an idea across these borders require a high level of internal permeability. *"Organization organized for better collaboration (=no filtering, no proxies, smaller proximity, time zone, etc. . .) is more likely to produce more innovative ideas. Layering, direct reporting, micro management, and similar old-school practices are killing innovation."*

Looking from the other perspective, new product innovations will create new demands and implications which will give rise for possibilities and triggers for organizational innovation [19]. *"New and exciting products means we have to adapt how we work to support these in the best-possible, not only from an engineering or software perspective, but for example from the launch projects etc."*

As has been discussed in regards to previous sections on the matter of product innovation versus process and business innovation, there exists a dual relationship here as well as exemplified by the response: *"Organizational innovation increases our capability to handle new and complex tasks. Innovative products will require us to handle new or more complex tasks and without room for growth, product innovation will fizzle."*

4.5 Product innovation challenges

In the responses, several aspects were mentioned as challenges to the product innovation.

a) *Idea tracing and execution uncertainty* – Even though there may be a rich pool of innovative ideas being produced and a general will to contribute, it is important to maintain and support it. Knowledge and awareness of what happens to ideas contributed to the innovation development process is important for the contributors to feel that they are taken seriously and that it is worth to continue contributing, which in turn gives an increased innovation capacity for the company [81]. When the ideas come bottom-up there needs to be a feedback loop top-down that stimulates this need of information as confirmed by Koc and Ceylan [82], and Wnuk et al. [149].

b) *Short term perspective* – By having a narrowed foresight, release planning tend to prioritize non-unique features which renders in low diversity in the product range, thus making the company being a follower of competitors rather than a leader. A longer time perspective needs to be integrated into the company culture, together with a positive mindset for game changers and innovative features to be created.

c) *Product scope and innovation strategy* – Uncertainty about the product roadmap and feature scope leads to risks that the creative minds of the company

are misdirected. A common and established innovation strategy can help defining the product scope and frame where ideas are needed suggested by Koc and Ceylan [82], and Wnuk et al. [149].

d) *Limiting environment and mindset* – Soft factors such as employees feeling that they can have a free mindset and share ideas openly is important for an innovative environment. It must be okay to test new ideas, but also to fail. These are factors, triggered by Ekvall's innovation climate model [41].

e) *Restriction by external stakeholders* – A commercial product company can have many stakeholders, some not being the end customer. This may include distributors and service providers further down the value chain, adding value and modifications to the product before they reach the final buyers. These stakeholders put requirements that may prevent and limit the feature scope possible to address. This filter risks to kill ideas inside the company and ignore needs, both identified and unidentified, from the end customers. This challenge is in line with Conboy and Morgan's findings [29].

f) *Limited time for innovation activities* – Tight project budgets and short deadlines are two factors that can restrict time available for idea creation. Developers usually have pet projects and ideas they would like to work on, some even dedicate their spare time for this purpose. By allowing the time, this can prove a valuable source of product innovation as suggested by Conboy and Morgan [29].

g) *Cross-functional resources* – Bringing new people together creates new product ideas and can boost innovation development. Cross-functional labs-sections and dedicated innovation team are two examples suggested by Conboy and Morgan [29], and Koc [81].

4.6 Process innovation challenges

This section presents the challenges, directly related to process innovations.

a) *Process change too slow* – The introduction of a new process may be cumbersome for several reasons, with the effect that the changes are implemented slowly. This can cause confusion for employees being caught between two states – before and after the change – and also result in an unsynchronized organization as different parts may adapt faster than others.

b) *Process change too often* – Another issue with respect to process change is that they may happen too often. This can be a cause effect relationship with an adoption process, as old processes risk being outdated once introduced if done in a too slow and inefficient manner. When the environment changes, for example technology and dependencies towards partner's progress, so does the requirements on the internal tools and processes have to change at the same pace. This can also relate to organizational innovation.

c) *Process change top down* – Problems can arise when a process is introduced top-down instead of bottom-up. Managers may not always know what is the most

efficient way to work compared to those actually performing the work. This challenge is also in line with the findings of Qin [116], and Wnuk et al. [149].

4.7 Business innovation challenges

Challenges related to business innovation are about alignment with the market and end users.

a) *Reaching the end-customers* – When there are layers between the producer and end-customer, for example, distributors and service providers, promotion of new ideas and product innovations to end-customers gets complicated. As technology and social habits evolve, new innovative ways are needed to keep pace with the different forums for communication used by the end-customers of today and tomorrow. Examples of such phenomena are software ecosystems [151].

b) *Product and marketing synchronization* – The views on what the top innovative features are may differ between different parts of the company. A misalignment like this can create confusion between marketing and product development. This could render in the wrong features being promoted. The suggested needs of the end customers should be communicated and synchronized to all relevant parts of the organization, e.g. product planning, marketing and development.

4.8 Organizational innovation challenges

Organizational innovation challenges relate to collaboration, communication and change.

a) *Closed organizational borders* – If the organization is too introvert and closed, opportunities, possible collaborations, sources of ideas and other possible inputs to their internal innovation process might be missed. By opening up the company borders for external collaboration and influence, new possibilities can arise both in regards to new innovations and markets, as described by the Open Innovation paradigm [24].

b) *Intra organizational collaboration* – Barriers and layers can prevent otherwise prosperous and potential collaborations between business units in organizations. Examples may be different sub-priorities of features between projects and multiple number of mangers creating a complex and bureaucratic hierarchy as identified among the respondents and confirmed by Koc [81]. These are related to what Bjarnason et al refer to as “gaps” [14]. Koc further points out that such cross-functional integration demands a high level of coordination, otherwise it will rather have a negative impact on the product innovation.

c) *Intra organizational learning* – Unawareness of what has been done in other parts of the company can create inefficiency and missed possibilities. In regards to process innovation, tools, technologies and processes from one part may prove its self superior or complementary to those used in other parts. And in regards to product innovation, a commoditized good or service from one business unit may

turn out as innovative if added to the value proposition in another business unit's product chain. This is a challenge in-common with inner source [94], but also one of the ways in how it can help organizations become more innovative by using it as a type of intra-organizational open innovation [101].

5 Conclusions

The view on what innovation is and where it can be performed is a diversified topic. OECD [4] differentiates between four types: product, process, market and organizational innovation. These were adopted in the survey on which this paper is based on, with a redefinition of market innovation into business innovation. The original definitions are general and applicable on a multiple number of fields. This paper puts them in the context of software engineering characterized by the opinions of people involved in different levels of a large software development organization.

The perception of the term *innovation*, to answer the first research question (See **RQ1**, Section 1), is diversified. Even though it is not general, some had trouble relating to the term innovation as such and when a feature or certain work can be classified accordingly. Some believed that they were not able to perform any innovative activities as it was not a part of their work description or role, which was present in connection to all of the four types of innovation. Apart from awareness and knowledge, another important factor that also needs consideration is the mindset of the employees that innovation is possible and something that they can help to create.

The different types cannot be considered isolated or decoupled which answers the second research question (See **RQ2**, Section 1). Connections between product innovation and process, business and organizational innovation exists in both directions. Introduction of product innovations creates demand and possibilities for processes, marketing and organization to adapt and optimize as the conditions has been changed. Interdependencies may require tailoring being done, either as a direct consequence or as a side effect. On the other way around, introduction of a process, business or organizational innovation can change the environment and conditions for how product development is being done. Inputs such as new technologies, ideas, resources and know-how are example factors which can be considered a cause behind a product innovation effect. Open innovation could be classified as an organizational innovation that can render inputs to the internal innovation process [24].

Challenges correlated to the different innovation types were also identified, with respect to the third research question (See **RQ3**, Section 1). These give a context to the term of innovation that covers parts other than the more normal conception of innovation in regards to just products. Some challenges may target more than one type of innovation, e.g. internal communication which can cause

issues for introduction on new processes and organizations as well as hinder ideas to be spread and discussed.

For future research it would be interesting with studies confirming and exemplifying the connections described, for example how process innovation could trigger product innovation. An anticipated challenge will be to trace a cause effect relationship and connecting the two areas. Another area also includes confirming the challenges identified, and further characterizing the innovation types from a software engineering perspective.

CHAPTER III

OPEN INNOVATION THROUGH THE LENS OF OPEN SOURCE TOOLS: AN EXPLORATORY CASE STUDY AT SONY MOBILE

Abstract

Context: Open Innovation (OI) is an emerging paradigm in software engineering and still little is known about what triggers software-intensive organizations to adopt it and how this affects SE practices. The OI model can be applied in numerous of ways, including organizations involvement in Open Source Software (OSS). Outcomes from the OI model are not restricted to product innovation but also includes process innovation, e.g. improved SE practices and tools.

Objective: This study explores the use of OSS and the involvement of Sony Mobile in OSS communities from an OI perspective. Furthermore, the study also highlights the innovative outcomes attached to OI participation and how SE practices have been adapted in relation to OI.

Research Method: An exploratory embedded case study design is used to target the objective in a real-world setting by investigating how Sony Mobile use and contribute to Jenkins and Gerrit, which are two central OSS tools in their continuous integration tool chain. Quantitative analysis is performed by extracting the change log data from source code repositories in order to identify the top contributors. Data from five semi-structured interviews are analyzed qualitatively to explore the nature of the contributions.

Conclusion: The findings of the case study include five major themes. i) The process of opening up towards the tool communities correlates in time with a general

adoption of OSS in the company. ii) Assets which are not competitive advantage nor a source of revenue are left open, and gradually, the company turns more and more open. iii) The requirements engineering process towards the community is informal and based on engagement. iv) The need for systematic and automated testing is still in its infancy, but the needs are identified. v) The innovation outcomes include the "free" features, maintenance and time, but increased speed and quality are also counted as OI outcomes.

1 Introduction

Organizations developing software-intensive products have recently been exposed to new facets of openness that go beyond their experience and provide opportunities outside their current ways of working. The shift from proprietary software engineering to Open Source Software (OSS) is well acknowledged and experienced by many companies, but the emerging new facets of openness stretch the transparent boundaries beyond the source code. Consequently, it leads organizations to consider openness in regards to their processes, business models and artifacts. Chesbrough [24] coined the term *open innovation* (OI) for this general phenomenon as *"a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology"*.

Increased openness poses significant challenges to software-intensive organizations in terms of securing their competitive advantage in relation to their competitors and thereby, challenges both on the operational and strategic levels. Specifically, as a continuously growing share of product and service innovation is implemented in software, the implications for software development is essential to explore. Existing research highlights different strategies such as open business models [21], organizations engaging in OSS communities [32] by adopting selective revealing [62] and values gained as a result of such involvement. However, it should be noted that OSS is *not* equivalent to OI in software-intensive organization, but only for certain cases an example of OI, and has so been for the past 20 years [22]. The main criteria for OSS to fit inside the OI model is that it is used in alignment with the company's business model and takes part in a generation of innovative outcomes, creating or capturing value for the company [24].

Companies intending to engage in OSS communities need to adapt their software development strategies to, for example, correcting bugs and actively participating in discussions and contributing new features back to the community. Subsequently, they might reduce maintenance cost compared to commercial software development, but require different modes of working [87, 121, 148] (process adjustment) and OSS governance mechanisms [90] to facilitate their software development once the organization decides to adopt OI. As consequence of adjusting their processes, companies are given better control over Research and Develop-

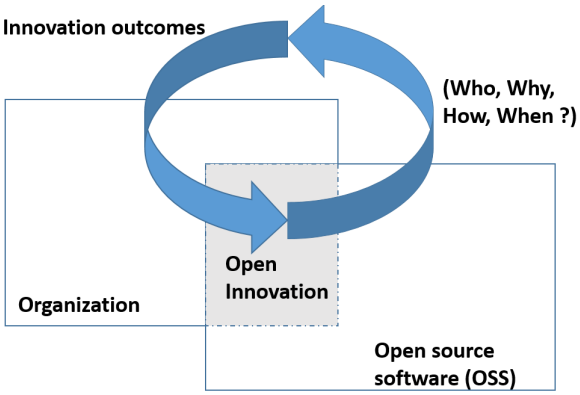


Figure 1: Study Objectives in the intersection between proprietary organizations and open source software.

ment (R&D) departments, how output is shared with its collaborators [16]. It is often discussed how OSS is a part of product offerings, either internal or external, or as a pool for R&D, with different strategies attached [132, 139].

However, existing literature does not go into detail reporting how OSS involvement may be utilized as an enabler and support for further innovation spread inside an organization, e.g. process or organizational innovations. Additionally, as has been shown in the intersection of OI and SE [107], little is known on how SE practices should be structured in order to support and optimize the output for a company involved in OI.

In this study, we present an exploratory case study [122] at Sony Mobile aiming to investigate OSS tools usage and involvement in the communities from an OI perspective. We seek to identify innovative outcomes and how SE practices have been adapted (see Fig. 1). The units of analysis are Jenkins and Gerrit, which are two central OSS tools in Sony Mobile's continuous integration tool chain. This study further investigates how value is captured externally via OSS tools development and how the gained value is transferred into the product development teams of the company (see Fig. 2).

We started by mining the Jenkins and Gerrit repositories, to characterize Sony Mobile's contributions, and to identify active participants for interviews. The found Interviewees were belonged to an internal Tools department, responsible for the development, support and maintenance of their continuous integration tool chain. Qualitative analysis of the interviews showed that the company had opened up as a consequence of a move from a proprietary to an OSS platform in their products. This had in turn set the company on a course towards a more open mindset with an OSS continuous integration tool chain and a continuous involvement in the related communities, causing a need of SE practices to be adjusted.

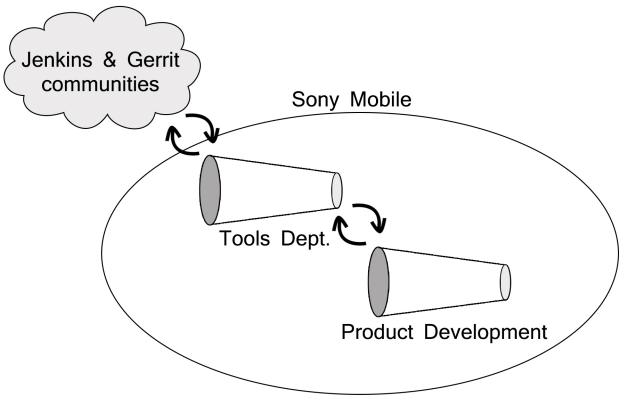


Figure 2: Study context

More explicitly, this study contributes by studying how OSS may be used to not only for leveraging product innovation in the tools themselves, but also how these tools can be used as enablers for process innovation in the form of improved SE practices and tools intra-organizationally.

This paper is structured as followed. Section 2 highlights the related work and Section 3 states the research methodology. In Sections 4 and 5 results from the quantitative and qualitative analysis are presented respectively. Finally, Section 6 explicate discussion in regards to the results followed by the conclusion in Section 4.

2 Related work

While there is a lack of studies on OI in software engineering [107], there is an abundant literature on the use of OSS in the development of proprietary software. OSS is used as an enabler for OI in an organization in order to create and capture value [26, 33, 62, 64]. Source code management repositories of large OSS projects can be a source of valuable data about the organizational structure, evolution, and knowledge exchange in the corresponding development communities [52, 98]. However, the volume of information that OSS offers requires efficient methods for highlighting the relevant information for a given aspect of the project. In order to extract the information from OSS code repositories, a tool called CVSAnalY [119] supports automatic and non-intrusive measurement and analysis, providing a real time and historical data about projects details and its contributors. Several studies have been conducted where OSS projects are analyzed by mining their source code change logs and mailing list archives to be able to understand the behavior of their respective communities [44, 52, 56, 111]. The studies either focus on an entire

portal, hosting hundreds of open source projects, or standalone projects [67]. The current study uses CVSAnalY to extract change log data for our unit of analysis, see Section 3.2. This lead to the identification of top stakeholders for the units of analysis in the study and identification of the key interviewees.

As far as the theoretical background for this study is concerned, Henkel et al. [62] studied 212 firms that manufactured embedded Linux drivers. Henkel et al. argued that OI is facilitated by Intellectual Property Rights (IPRs) and selective revealing of IPRs may be beneficial for their business. In addition, one of the biggest triggers for revealing IPRs was the customer demand. Moreover, in another case study of embedded Linux Henkel et al. [139] found that developers associated with universities and hobbyists reveal all their code in contrast to many other firms. On the other hand, Heck et al. [61] investigated the requirements engineering process in OSS projects websites and found that issue trackers are used to communicate requirements. Their idea was to improve the tool support for dealing with feature requests in issue trackers and give users of these issues trackers an overview of the project, including relationships between already existing feature requests. Höst et al. [66] conducted a study to understand how large organizations collaborate on the Android project. The results indicate that the project is highly influenced by Google's development effort. In addition, Alexy et al. [6] studied the impact of individuals on OSS adoption. In terms of technical dimensions, findings suggest that giving roles by management are less effective than taking roles (volunteers) since OSS is a novel process of innovation in software development.

Krogh at al. [136] investigated the strategies and processes by which new people (individual level OI) join the existing community of software developers, and how they initially contribute code. The study developed a construct entitled *joining script*, and proposed that contributors who follow joining scripts (offer bug fix, report bugs, discussion, feedback etc.) are more likely to obtain access to the community. Consequently, a developer is granted access to a privileged source code commit regime. Talking about OI at individual level, Dahlander [34] claimed that initiating OSS projects is often a good way of attracting the most pragmatic developers from the communities. Moreover, having an employee in the community seems to be the key enabler for the firms to not only gain a good reputation but also to influence the direction of the development towards the firms' own interests. West et al. [144] examined the complex ecosystem surrounding Symbian Ltd. and identified three inherent difficulties for firms leading an OI ecosystem: 1) prioritizing the conflicting needs of heterogeneous ecosystem participants, 2) knowing the ecosystem requirements for a product that has yet to be created, and 3) balancing the interests of those participants against those of the ecosystem leader.

West and Gallagher [139] mention strategies that firms deploy to generate external knowledge, incorporating the external innovation into firms capabilities and exploiting the IPRs by selective revealing. These strategies are comprised of pooled R&D, spin-outs, selling complement and donating complements. Linden et al. [96] argued that software products lose value with the passage of time due to

ever growing and improving software components, and thereby products become a good candidate for OSS development. Stuermer et al. [132] conducted a study on implementing a private collective model at Nokia to identify the incentives for firms and individuals in investing OSS. The study examined the development of the Nokia Internet Tablet which builds on a hybrid of OSS and proprietary software development. The results indicated that cost of the model in terms of difficulty to differentiate, guarding business secrets, reducing the community barriers and giving up organizational control. On the other hand, Nokia reaped benefits in terms of low knowledge protection costs, learning effects, reputation gain, reduces development and innovation costs.

Given the scarce literature on OI in SE, it remains unclear what triggers software-intensive organizations to open up and what are the key factors organizations consider before opening up a project. Moreover, it also needs to be investigated whether or not the existing requirements engineering and testing process of SE are fit enough to deal with the challenges of OI. Finally, the study also highlights the innovation outcomes attained as a result of the openness.

3 Case study design

The aim of this study is to explore how large software-intensive organizations use and develop OSS tools in support of their product development. The objective is then to investigate how product innovation is captured in the tools and how this can be used as enablers for process innovation in the form of improved SE practices and tools intra-organizationally. The case company is Sony Mobile and the units of analysis are Gerrit [3] and Jenkins [2], both OSS tools part of their continuous integration tool chain.

1. ***Gerrit*** is an OSS code review tool created by Google in connection with Android in 2007. It is tightly integrated with the software configuration management tool GIT, working as a gatekeeper, i.e. a commit needs to be reviewed and verified before its allowed to be merged into the main branch.

2. ***Jenkins*** is an open source build server that runs on a standard servlet container e.g. Apache TomCat. It can handle Maven and Ant instructions, as well as execute custom batch and bash scripts. It was forked from the Hudson build server in 2010 due to a dispute between Oracle and the rest of the community.

3.1 Research questions

All research questions are formulated to study the OI phenomenon of Sony Mobile in an exploratory manner. To be more specific, we are investigating how Sony Mobile use OSS in their continuous integration process to enable OI. As presented

Table 1: Research questions with description

Research Questions	Objective
RQ1: How and to what extent is Sony Mobile involved in the communities of Jenkins and Gerrit?	To characterize Sony Mobile's involvement and identify potential interviewees.
RQ2: What is the motivation for Sony Mobile to adopt an OI approach?	To explore the transition from a closed innovation process to an OI process of Sony Mobile
RQ3: How is the process behind the decision to go open with a project or feature structured?	To investigate what factors affect the decision process when determining whether or not the Sony Mobile should contribute functionality.
RQ4: What are the innovation outcomes as a result of OI participation?	To explore the vested interest of Sony Mobile as they moved from a closed innovation model to an OI model
RQ5: How do the requirements engineering and testing processes interplay with their OI adoption?	To investigate the Requirements Engineering and Testing process and how they deal with the special complexities and challenges involved due to OI.

in the figure 2, we started investigating the OI phenomena by identifying the the key contributors of Jenkins and Gerrit, and that lead to a more focused exploration of how, when and what triggers Sony Mobile to utilize OI. Most importantly, what innovation outcomes Sony Mobile gain as a result of adopting OI. *RQ1* addresses the extent to which Sony Mobile is involved in the Jenkins and Gerrit community and its key contribution areas (i.e. bug fixes, new features, documentation. etc.). *RQ2* and *RQ3* explore the rationale behind Sony Mobile's transition from closed innovation to OI. Furthermore, *RQ4* highlights the key innovation outcomes realized as a result of the openness. The final research question (*RQ5*) aimed at understanding whether or not the existing requirements engineering and testing processes have the capacity to deal with the OI challenges in SE. All research questions under investigation are stated in Table 1 with their objectives. *RQ1* was answered with the help of the quantitative analysis of repository data, while the remaining four research questions (*RQ2, RQ3, RQ4, RQ5*) were investigated using qualitative analysis of interview data.

3.2 Case selection and Units of analysis

Sony Mobile is a multinational corporation with roughly 5,000 employees globally, developing embedded devices. The studied branch is focused on development of Android based devices. Moreover, the branch under investigation is comprised of 1,600 employees and 900 of them are involved in the development of these communication devices. Sony Mobile develops software in an agile fashion and uses software product line management with a database of more than 20,000 features suggested or implemented across all product lines [114].

The continuous integration tool chain used by Sony Mobile is developed, maintained and supported by an internal Tools department. The teams working on Sony Mobile's core products are thereby relieved of this technical overhead. During the later years, it has been recognized by Sony Mobile that they are becoming more open in the sense that they mature in usage and involvement in OSS communities. This maturity can be compared to going from a closed innovation model to OI model [24], given that business values are created or captured as an effect.

From an OI perspective, there are interactions between the Tools department and the communities surrounding Jenkins and Gerrit (see Fig. 2). The in- and outgoing transactions, visualized by the arrows, are data and information flows, e.g. ideas, support and contributions, can be termed as a coupled innovation process [42]. The exchange is continuous and bi-directional, and brings product innovation into the Tools department in the form of new features and bug fixes to Jenkins and Gerrit.

The Tools department can in turn be seen as a gate between external knowledge and the other parts of Sony Mobile (see Fig. 2). The purpose of the Tools department is to access, adapt and integrate the knowledge they obtain externally from Jenkins and Gerrit, into the product development teams of Sony Mobile. This creates another set of transactions inside the domain of Sony Mobile which can be labeled as process innovation [4] in the sense that new tools and adapted ways of working bring a higher quality and efficiency to the development process. This relates to the need of internal complementary assets that is mentioned as an area for future research by Chesbrough et al. [22]. Therefore, Sony Mobile is chosen as a suitable case for OI in software engineering.

We utilized the case study design with Jenkins and Gerrit as units of analysis [122] as these are the products in which the exchange of data and information enable further innovation inside Sony Mobile, see Fig. 3.

3.3 Case study procedure

We performed the following steps (see Fig. 4).

1. Preliminary investigation of Jenkins and Gerrit repos.

2. Mine the identified project repositories.

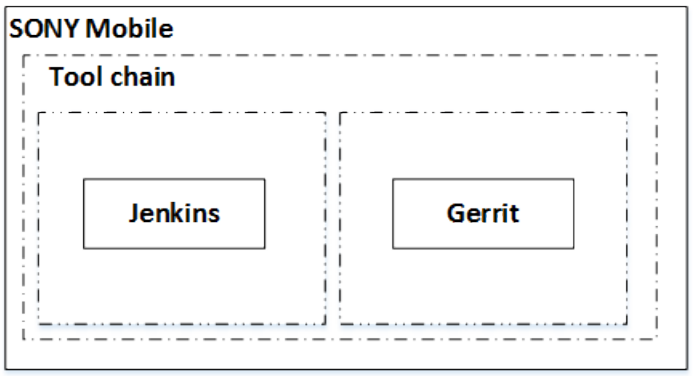


Figure 3: Units of analysis

3. Extract the change log data from the source code repositories.
4. Analyze the change log data (i.e. stakeholders, commits etc).
5. Prepare and conduct semi-structured interviews to answer *RQ2, RQ3, RQ4* and *RQ5*.
6. Summarize the findings from the change log data to answer *RQ1*.
7. Data synthesises
8. Answers to *RQ1, RQ2, RQ3, RQ4* and *RQ5*.

3.4 Methods for quantitative analysis

To get an initial understanding of Sony Mobile's involvement in the Open Source tools (*RQ1*), we started with quantitative analysis of commit data in the source code repositories of Jenkins and Gerrit.

Preliminary investigation of Jenkins and Gerrit

During the initial screening, we mined the source code repositories of Jenkins and Gerrit using a unified procedure. After cloning the repositories locally, we extracted the commit id, date, contributor name, contributor email and commit description message for each commit, with the help of the CVSAnly tool [119]. The extracted data was stored in a relational database which scheme is standardized in the tool, independently of which source code repository is analyzed.

The structure of the database allows a quantitative analysis to be done by writing specific SQL queries. To get an overview of the contributors to the communities, the number of commits per contributor were added together with the name and

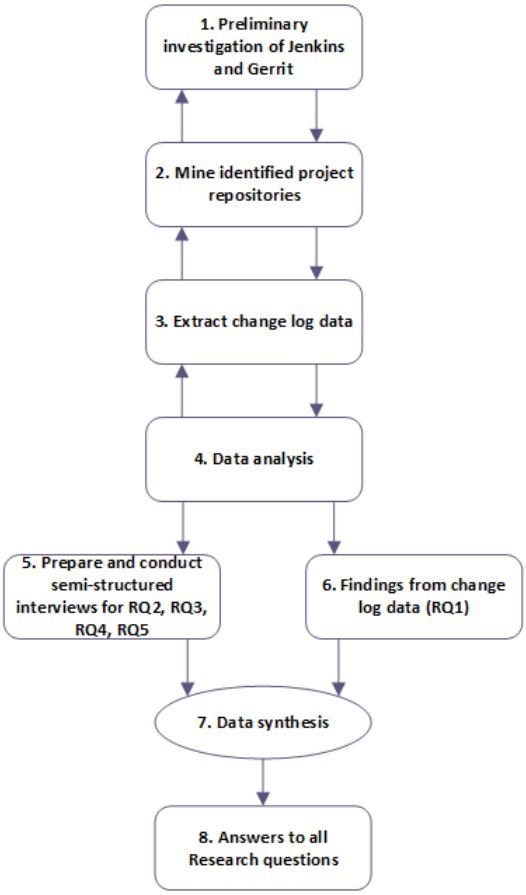


Figure 4: Case study procedure

email of the contributor as keys. We extracted the affiliations of the contributors from their email addresses.

Sony Mobile turned out to be one of the main contributors to Gerrit while limited evidence of contributions to the Jenkins community was identified. The reason for this was that Jenkins is a plug-in-based community, i.e. there is a core component and then about 1,000 plug-ins of which each has a separate source code repository and community. Our initial screening had only covered the core component. After analyzing forum postings, blog posts and reviewing previously identified contributors on Ohloh.net[1] and LinkedIn[2], a set of Jenkins plug-ins, as

[1] https://www.openhub.net/
[2] https://www.linkedin.com

well as two Gerrit plug-ins, were identified which then were screened with the same process as applied earlier.

The following Open Source projects are included for further analysis.

- Gerrit
- PyGerrit (Gerrit plug-in)
- Gerrit-Events (Gerrit plug-in)
- Jenkins-Gerrit trigger (Jenkins plug-in)
- Build-Failure-Analyzer (Jenkins plug-in)
- External resource viewer (Jenkins plug-in)
- Team views (Jenkins plug-in)

Classification of commit messages

Further quantitative analysis was applied to the Jenkins and Gerrit commit data available through the databases created in section 3.4. Smaller amounts of commits were re-assigned to a different organization as these could be connected to earlier identified contributors with non-affiliated email addresses. After this, the list of top contributors were compiled as well as a time series analysis with commits added together per year in order to see how Sony Mobile's involvement evolved. As a next step, we characterized the contributions made by Sony Mobile to the communities. Afterwards, all issues (bugs and feature requests) available for Gerrit on the community's online issue trackers were collected into a comma separated file. The ones that could be traced to Sony Mobile were filtered out. Each issue has a unique id which was used as a key and searched for in the commit description messages. This filtration however only rendered in a smaller subset of commits, not close to the total number, which had been previously identified.

As an alternative solution we manually analyzed the commit description messages. The criteria mentioned by Hattori et al. [60] were used to classify the commits made by Sony Mobile. The classification is as follows:

Forward engineering activities refer to the incorporation of new features and implementation of new requirements including the writing new test cases to verify the requirements.

Re-engineering activities deal with re-factoring, redesign and other actions to enhance the quality or the code without adding new features.

Corrective engineering activities refer to handling defects, errors and bugs in the software.

Management activities are related to code formatting, configuration management, cleaning up code and updating the documentation of the project.

Multiple researchers were involved in the commit message classification process. After defining the classification categories, Kappa analysis was performed to calculate the inter-rater agreement level. First, a random sample of 34% of total commit messages were taken to classify the commit messages and Kappa was calculated to be 0.29. Consequently, disagreement were discussed and resolved since the inter-rater agreement level was below substantial agreement range. Afterwards, Kappa was calculated again and found to be 0.94.

3.5 Methods for qualitative analysis

The quantitative analysis along with archival analysis of related forums, blog posts and community related online sources had given a solid foundation to understand the relation between Sony Mobile and Jenkins and Gerrit communities. Therefore, in the next step we added a qualitative view by interviewing relevant people inside Sony Mobile in order to address *RQ2–RQ5*.

Interviewee selection

The selection of interviewees was based on the contributors identified in the initial screening of the projects. Three candidates were identified and contacted by mail (Interviewees 1-3, see Table 1). Interviewees 4-5 were proposed during the initial three interviews. The first three are top contributors to the Jenkins and Gerrit communities, giving the view of Sony Mobile's active participation and involvement with the communities. It should be noted that I3, when (s)he was contacted, had just left Sony Mobile for a smaller company dedicated to Jenkins development. (S)he held his position as a tools manager for Jenkins was filled later by an I1. I4 is a Software Architect in the Tools department involved further down in Sony Mobile's continuous integration tool chain and gives an alternative perspective on the OSS involvement of the Tools department as well as a higher, more architectural view of the tools. I5 is an upper-level manager responsible for Sony Mobile's overall OSS strategy, which could contribute with a top-down perspective to the qualitative analysis.

Analysis procedure

The interviews were semi-structured, meaning that interview questions were developed in advance and used as a frame for the interviews, but still allowing the interviewers to dig deeper into findings during the interview wherever needed. The two first authors were present during all five interviews, with the addition of the third author during the first and fifth ones. Each interviewer took turns asking questions, whilst the others observed and took notes. Each interview was recorded and transcribed. A summary was also compiled and sent back to the interviewees for a review. Any misunderstandings or corrections could then be sorted out.

Table 2: Interviewees demographics

Anonymous name	ID	Tools involvement	Years of experience	Role
Interviewee 1	I1	Jenkins	8 Years	Tools manager for Jenkins
Interviewee 2	I2	Jenkins and Gerrit	6 Years	Team lead, Tools manager for Gerrit
Interviewee 3	I3	Jenkins	7 Years	Former tools manager Jenkins
Interviewee 4	I4	Second line after Jenkins and Gerrit Build artifacts and channel distribution	8 years	Software Architect
Interviewee 5	I5	Open Source policy in general	20+ Years	Upper-level manager responsible for overall Open Source strategy

The interview findings were discussed and if relevant, interview questions were updated for upcoming interviews.

The two first authors individually conducted the qualitative analysis using a thematic analysis approach [30,31] followed by a review from the third, fourth and fifth authors in the study. This resulted in two sets of themes and characteristics based on the five interviews. These were then summarized under five main themes (see Table 3), which is presented in the results section.

3.6 Validity Threats

This section highlights the validity threats related to the case study. Four types of validity threats [122] are addressed with their mitigation strategies.

Internal Validity

Internal validity concerns casual relationships and is primarily used for explanatory studies. It was found out in the study that Sony Mobile does not use any general innovation metrics. Therefore researchers had to rely on qualitative data to infer the possible unconsciously used metrics. This leads to the risk of introducing an element of subjectivity from researchers while inferring innovation outcomes as a result of OI adoption. In order to minimize this risk, the first two authors performed the analysis independently and the remaining authors reviewed it to make

the synthesis more objective. Moreover, findings were sent back to interviewees to validate the findings drawn from qualitative data.

External Validity

The external validity refers to the extent it is possible to generalize the study findings to other contexts. The scope of this study was limited to software-intensive organization utilizing the notion of OI to accelerate their innovation process. The selected case company is a large scale organization with an intense focus on software development for embedded devices. Moreover, Sony Mobile is a direct competitor of all the main stream companies making Android phones. Like Sony Mobile, the involvement of all stakeholders in the units of analysis (Jenkins and Gerrit) indicate their adoption of Google's tool chain to improve their continuous integration process. Therefore, the findings of this study may be generalized to major stakeholders identified for their contributions to Jenkins and Gerrit.

Construct Validity

Construct validity deals with choosing the right measures for the concepts under study. In this study this relates to the selection of interviewees. First we conducted a preliminary quantitative analysis of the repositories before selecting the interviewees. In addition, the selection was performed based on the individuals' contributions to the units of analysis (Jenkins or Gerrit). Some of the key factors considered before selecting interviewees were process knowledge, role, and visible presence in the community. Second, the presence of researchers at Sony Mobile was a threat that could affect the outcome of the study. This threat was limited as there has been a long research collaboration of between the university and Sony Mobile. Third, multiple researchers validated the interview questionnaire followed by a pilot interview in order to avoid misinterpretation of interview questions. Finally, in order to avoid subjectivity, the authors opted for the objective criterion proposed by Hattori et al. [60] to classify the commits messages. During the discussion, the disagreements were identified using Kappa analysis and resolved to achieve a substantial agreement.

Reliability

The reliability deals with the ability to replicate the same study with same results in other settings. There is always a risk of introducing subjectivity from researchers. To mitigate this risk, multiple researchers individually transcribed and analyzed the interview data to make the findings more objective. In addition, multiple data sources were considered in the study followed by quantitative and qualitative analysis to ensure the correctness of the findings. All interviews were recorded, transcribed and sent back to interviewees for validation. The commit database analysis was highly automated and therefore the subjectivity risks are

minimized. Finally, this study was not performed to get an approval from Sony Mobile to explore its OI activities instead, the idea was to to keep the study design and findings as transparent as possible without making any adjustments in the data except for the anonymous interviewees. Later on, results were shared with Sony Mobile.

4 Quantitative analysis

This section presents the classification of Sony Mobile's commit messages (see Fig. 5) and time series analysis (see Fig. 6).

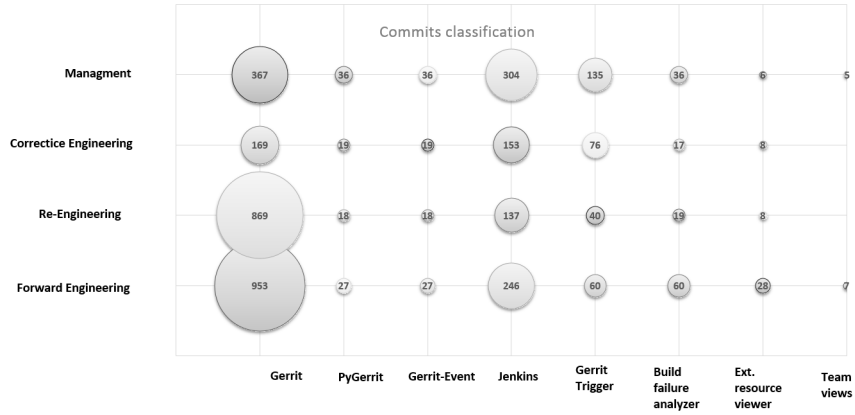


Figure 5: Bubble plot for Sony Mobile commits by classification

4.1 Gerrit

The time series analysis of the Gerrit commit data indicates that a large portion of the commits, generated by Sony Mobile was made during 2012, which is subject to further investigation (see Fig. 6)

PyGerrit

Pygerrit is a Python library that provides a way for clients to interact with Gerrit Code Review via ssh or the REST API [3]. As can be seen in Fig. 4, Sony Mobile initiated this plugin and is the biggest contributor to it. Time series analysis shows

[3]REST (REpresentational State Transfer) is a simple stateless architecture that generally runs over HTTPS/TLS. The REST style emphasizes that interactions between clients and services are enhanced by having a limited number of operations

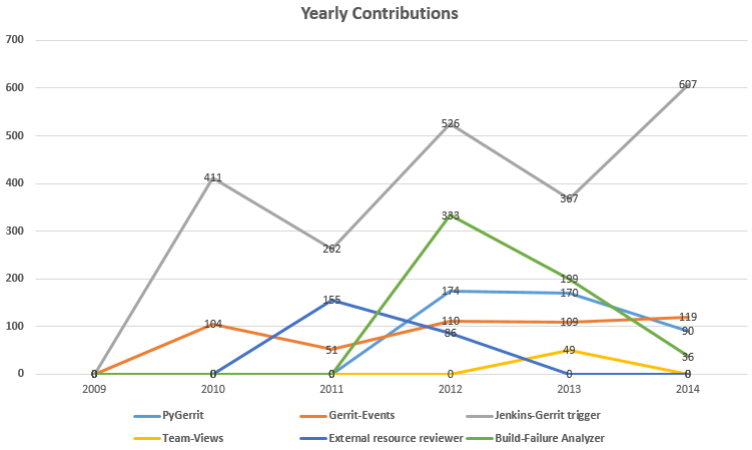


Figure 6: Time series analysis of Sony Mobile's contributions

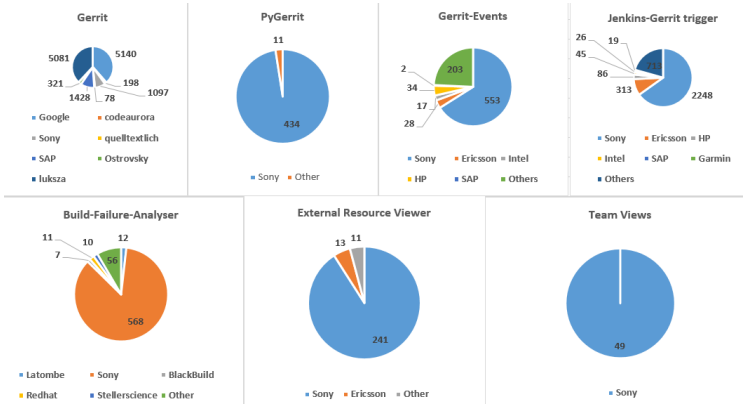


Figure 7: Analysis of Sony Mobile's contributions.

a steady growth of Sony Mobile's contributions since its introduction from 2011 on-wards (see Fig. 6)

Gerrit-Event

Gerrit-Event is a Java library used primarily to listen to stream-events from Gerrit Code Review and to send reviews via the SSH CLI or the REST API. It was originally a module in the Jenkins Gerrit Trigger plug-in and is now broken out to be used in other tools without the dependency to Jenkins. Apart from Sony Mobile, HP, SAP, Ericsson and Intel contributions reveal that they are also using Gerrit-Event in their continuous integration process. Sony Mobile started contributing to Gerrit-Event in 2009 and since then seem to be the largest contributor along with its competitors (see Fig. 6).

4.2 Jenkins

Contributions from Sony Mobile to Jenkins could not be identified in the core product but to a various set of plug-ins. The ones identified are:

- Gerrit Trigger-plug-in
- Build-Failure-Analyzer-plug-in
- External resource-dispatcher-plug-in
- Team-views-plug-in

Gerrit Trigger

This plug-in triggers builds on events from the Gerrit code review system by retrieving events from the Gerrit command "stream-events", so the trigger is pushed from Gerrit instead of pulled as scm-triggers usually are. Multiple builds can be triggered by one change-event, and one consolidated report is sent back to Gerrit. This plug-in (see Fig. 6) seem to attract the most number of contributions from Sony since 2009 with most number of contributions (607) in the year 2014.

Build failure analyzer

This plug-in scans build logs and other files in the workspace for recognized patterns of known causes to build failures and displays them on the build page for quicker recognition of why the build failed. As can be seen in Fig. 6, Sony Mobile came out as the largest contributor in build failure analyzer. Moreover, commits contribution have declined after 2012 and one possible explanation for this is that it could be very specific to the needs of Sony Mobile since they are by far the largest contributor(see fig. 4).

External resource viewer

This plug-in adds support for external resources in Jenkins. An external resource is something external attached to a Jenkins slave and can be locked by a build, which thus gets exclusive access to it, then released after the build is done. Examples of external resources are phones, printers and USB devices.

Team views

This plug-in provides teams, sharing one Jenkins master, to have their own area with views similar to a User's my-views. The least number of commits were found in the team views with Sony Mobile turned out to be the only contributor for this tool (see Fig. 4).

5 Qualitative analysis

In order to analyze the qualitative data collected through interviews, we conducted thematic analysis [30, 31] to find recurring patterns. The following steps were performed in the process.

1. Extract data from interviews through transcription
2. Identify the common themes in the data
3. Define and group themes into distinct categories
4. Summarize findings

This resulted in five themes listed in Table 3, which are further presented in the following subsections.

5.1 Opening up

This theme is related to *RQ2*, see Table 3. The process of opening up for external collaboration and maturing as an open source company, can be compared to moving from a closed innovation model to an OI model [22]. The data suggest that the trigger for this process was a paradigm shift around 2010 when Sony Mobile moved from the Symbian platform (developed in a joint venture), to Google's open source Android platform in their products [145]. Switching to Android correlates to a general shift in the development environment, moving from Windows to Linux. This concerned the tools used in the product development as well. A transition was made from existing proprietary solutions, e.g. the build-server Electric commander, to the tools used by Google in their Android development, e.g. GIT and Gerrit. As stated by I2, *"... suddenly we were almost running pretty much everything, at least anything that touches our phone development, we were running*

Table 3: Thematic analysis

Theme name	Definition
Opening up (RQ2)	Sony Mobile's transition process from closed innovation model to OI model
Determinants of openness (RQ3)	Factors that Sony Mobile considers before indulging themselves into OI
Requirements engineering (RQ5)	How Sony Mobile manages their requirements while working in OI context.
Testing process (RQ5)	How Sony Mobile manages their testing process while working in OI context
Innovation outcome (RQ4)	The outcomes for Sony Mobile as a consequence of adopting OI

on Linux and open source". This was not a conscious decision from management but rather something that grew bottom-up from the engineers. The engineers further felt the need for easing off the old and complex chain of integration and building process.

At the same time, a conscious decision was made regarding to what extent Sony Mobile should invest in the open source tool chain. As stated by I5, *"... not only should [the tool chain] be based on OSS, but we should behave like an OSS good citizen and start to contribute, in the ways we can control it, understand it and even steer it up to the way we want to have it"*. This vision was concurrent with the internal paradigm shift at Sony Mobile and demanded organizational and process changes from both self-emergent bottom-up and top-down directions. The biggest hurdle concerned the notion of giving away internally developed IP rights, which could represent competitive advantages. The legal department took time in understanding the benefits and license aspects, which caused the initial contribution process to be extra troublesome. In this case, Sony Mobile benefited of having an internal champion and OSS evangelist (I5). (S)he helped to drive the initiative from the management side, explained the issues and clarified concerns from different functions and levels inside Sony Mobile. Another obvious success factor was the creation of an OSS review board, which included different stakeholders such as legal department representatives, user experience (UX) design, product development and product owners. This allowed for management, legal and technology representatives to meet in an open forum. The OSS contribution process now includes submitting a form for a review, which promotes it further after successful initial screening. Next, the OSS review board gives it a go or no-go decision. As this would prove bureaucratic if it would be needed for each and every contribution to an OSS community, frame-agreements are created for open source projects with a high-intensity involvement, e.g. Jenkins and Gerrit. This creates a simplified and

more sustainable process allowing for a day to day interaction between developers in the Tools department and the communities surrounding Jenkins and Gerrit.

5.2 Determinants of openness

This theme is related to *RQ3*, see Table 3. Several factors interplay in the decision process of whether or not a feature or a new project should be made open. Jenkins and Gerrit are neither seen as a part of Sony Mobile's competitive advantage nor as a revenue source. This is the main reason why developers in the Tools department can meet with competitors, go to conferences, give away free work etc. This, in turn, builds a general attitude that when something is about to be created, the question asked beforehand is if it can be made open source. There is also a follow-up question, whether Sony Mobile would benefit anything from it, for example maintenance, support and development from an active community. If a feature or a project is too specific and it is deemed that it will not gain any traction, the cost of generalizing the project for open use is not motivated. Another factor is whether there is an existing community for a feature or a project. By contributing a plug-in to the Jenkins community or a feature to Gerrit there is a chance that an active workforce is ready to adopt the contribution, whilst for new projects this has to be created from scratch which may be cumbersome.

Another strategic factor concerns having a first-mover advantage. Contributing a new feature or a project first means that Sony Mobile as the maintainer gets a higher influence and a greater possibility to steer it in their own strategic interest. If a competitor or the community publishes the project, Sony Mobile may have a lesser influence and will have to adapt to the governance and requirements from the others. A good example here is Gerrit Trigger. The functionality was requested internally at Sony and therefore undergone development by the Tools department during the same period, it became known that there was similar development ongoing in the community. As stated by I3, *"... we saw a big potential of the community going one way and us going a very different route"*. This led to the release of the internal Gerrit Trigger as an open source plug-in to Jenkins, which ended up being the version with gained acceptance in the Jenkins and Gerrit communities. The initial thought was however to keep it closed according to I3, *"... We saw the Gerrit trigger plug-in as a differentiating feature meaning that it was something that we shouldn't contribute because it gave us a competitive edge towards our competitors [in regards to our continuous integration process]"*. It should be noted that this was in the beginning of the process of opening up in Sony Mobile and a positive attitude was rising. A quote from I3 explains the positive attitude of the organization which might hint about future directions, *"... In 5 years time probably everything that Sony Mobile does would become open"*.

5.3 Requirements engineering process

This theme is related to *RQ5* (see Table 3) and provides insights about requirements engineering practices in an example OI context. The requirements process in the Tools department towards the Jenkins and Gerrit communities does not seem "very rigid" as is a common characteristic for OSS [124]. The product development teams in Sony Mobile are the main customers of the Tools department. These are, however, quite silent with the exception of one or two power users. There is an open backlog for internal use inside Sony Mobile where anyone from their product development may post feature requests. However, a majority of the feature requests are submitted via e-mail. The developers in the Tools department started arranging monthly workshops where they invited the power users and the personnel from different functional roles in the product development organization. An open discussion is encouraged allowing for people to express their wishes and issues. An example of an idea sprung out from this forum is the Build Failure Analyzer[4] plug-in. Most of the requirements are, however, elicited internally within the Tools department in a dialogue between managers, architects and developers. They are seen to have the subject matter expertise in regards to the tool functionality. According to I2, there are *"... architect groups which investigate and collaborate with managers about how we could take the tool environment further"*. This is formulated as focus areas, and *"... typical examples of these requirements are sync times, push times, build times and apart from that everything needs to be faster and faster"*. These requirements are high level and later delegated to the development team for refinement.

The Tools team works in an agile Scrum-like manner with influences from Kanban for simpler planning. The planning board contains a speed lane which is dedicated for severe issues that need immediate attention. The importance of being agile is highlighted by I2, *"... We need to be agile because issues can come from anywhere and we need to be able to react"*.

The internal prioritization is managed by the development team itself, on delegation from the upper manager, and lead by two developers which have the assigned role of tool managers for Jenkins and Gerrit respectively. The focus areas frame the areas which need extra attention. Every new feature is prioritized against existing issues and feature requests in the backlog. External feature requests to OSS projects managed by the Tools department (e.g. the Gerrit Trigger plugin) are viewed in a similar manner as when deciding whether to make an internal feature or project open or not. If it is deemed to benefit Sony Mobile enough it will be put in the backlog and it will be prioritized in regards to everything else. As stated by I3, *"... We almost never implemented any feature requests from outside unless we think that its a good idea for [Sony Mobile]"*. If it is not interesting enough but still a good idea, they are open for contributions from the community.

[4] https://wiki.jenkins-ci.org/display/JENKINS/Build+Failure+Analyzer

An example regards the Gerrit Trigger plugin and the implementation of different trigger styles. Pressing issues in the Tools department backlog kept them from working on the new features. At the same time, another software intense organization with interest in the plug-in contacted the Tools department about features they wanted to implement. These features and the trigger style functionality required larger architectural reconstruction. It was agreed that the external organization would perform the architectural changes with a continuous discussion with the Tools department. This allowed for a smaller workload and possibility to implement this feature earlier. This feature-by-feature collaboration is a commonly occurring practice as highlighted by I1, *"It's mostly feature per feature. It could be "[Company] wants this feature and then they work on it and we work on it". But we don't have any long standing collaborations"*. I3 elaborates on this further and states that *"... its quite common for these types of collaboration to happen just between plugin maintainer and someone else. They emailed us and we emailed back"* as was the case in the previous example.

In the projects where the Tools department is not a maintainer, community governance needs more care. In the Gerrit community, new features are usually discussed via mailing lists. However, large features are managed at hackathons by the Tools department where they can communicate directly with the community to avoid getting stuck in tiny details [101]. As brought up by I2, *"... with the community you need to get people to look at it the same way as you do and get an agreement otherwise it will be just discussions forever"*. This is extra problematic in the Gerrit community as the inner core team with the merge rights consists of only six people, of which one is from Sony Mobile. One of the key features received from the community was the tagging support for patch sets. I2 stated, *"... When developers upload a change which can have several revisions, it enabled us to tag meta-data like what is the issue in our issues handling system and change in priorities as a result of that change. This tagging feature allows the developers to handle their work flow in a better way"*. This whole feature was proposed and integrated during a hackathon, and contained more than 40 shared patch sets. Prior to implementing this feature together with the community (I3 quoted) *"... we tried to do it with the help of external consultants but we could not get it in, but meeting core developer in the community did the job for us"*. This explains yet another reason for Sony Mobile to open up [70].

As hackathons may not always be available, an alternative way to communicate feature suggestions more efficiently is by mock-ups and prototypes. I3 described how important it is to sell your features and get people excited about it. Screenshots is one way to visualize it and show how it can help end-users. In the Jenkins community, this has been taken further by hosting official web casts where everyone is invited to present and show new development ideas. Apart from using mailing lists and existing communication channels, Sony Mobile creates their own channels, e.g. with public blogs aimed at developers and the open source communities.

This close collaboration with the community is important as Sony Mobile does not want to end up with an internal fork of any tool. An I2 quoted, *"If we start diverging from the original software we can't really put an issue in their issue tracker because we can't know for sure if it's our fault or theirs system and we would lose the whole way of getting help from community to fix stuff and collaborate on issues"*. Another risk would be that *"...All of sudden everybody is dependent on stuff that is taken away from major version of Gerrit. We cannot afford to re-work everything"*. Due to these reasons, the Tools department is keen on not keeping stuff for themselves, but contributing everything [134, 148]. An issue in Jenkins is that there exists numerous combinations and setting of plug-ins. Therefore, it is very important to have backward compatibility when updating a plug-in and planning new features.

5.4 Testing process

This theme highlights the testing process aspects associated with *RQ5*, see Table 3. As with the requirements process, the testing process does not seem very rigid either. I3 quoted, *"...When we fix something we try to write tests for that so we know it doesn't happen again in another way. But that's mostly our testing process I think. I mean, we write JUnit and Hudson test cases for bugs that we fix"*.

Bugs and issues are, similarly to feature requests, reported internally either via e-mail or an open backlog. Externally, bugs or issues are reported via the issue trackers available in the community platforms. The content of the issue trackers is based on the most current pressing needs in the Tools department. Critical issues are prioritized via the Kanban speed lane which refers to a prioritized list of requirements/bugs based on the urgent needs of Sony Mobile. If a bug or an issue has low priority, it is reported to the community. This self-focused view correlates with the mentality of how the organization would benefit from making a certain contribution, which is described to apply externally as well, *"...Companies take the issues that affect them the most"*. However, it is important to show to the community that the organization wants to contribute to the project as a whole and not just to its parts, as mentioned by Dahlander [34]. In order to do so, the Tools department continuously stays updated about the current bugs and their status. It is a collaborative work with giving and taking. *"Sometimes, if we have a big issue, someone else may have it too and we can focus on fixing other bugs so we try to forward as many issues as possible"* [63].

In Gerrit, the Tools department is struggling with an old manual testing framework. Openness has lead them to think about switching from the manual to an automated testing process. I2 stated, *"...It is one of my personal goals this year to figure out how we can structure our Gerrit testing in collaboration with the community. Acceptance tests are introduced greatly in Gerrit too but we need to look into and see how we can integrate our tests with the community so that the whole testing becomes automated"*. In Jenkins, one of the biggest challenges in regards

to test is to have a complete coverage as there are many different configurations and setups available due to the open plug-in architecture. However, Gerrit still has some to catch up as stated by I2, *"it is complex to write stable acceptance tests in Gerrit as we are not mature enough compared to Jenkins"*. A further issue is that the test suites are getting bigger and therefore urges the need for automated testing.

Jenkins is considered more mature since the community has an automated test suite which is run every week when a new version of the core is released. This test automation is very recent. Selenium, which is an external OSS test framework, is used to facilitate the automated acceptance tests. It did not get any traction until recently because it was written in Ruby, while the Jenkins community is mainly Java-oriented. This came up after a discussion at a hackathon where the core members in the community had gathered, including representatives from the Tools department. It was decided to rework the framework to a Java-based version, which has helped the testing to take off although there still remains a lot to be done.

I3 highlighted that Sony Mobile played an important part in the Selenium Java transition process, *"The idea of an Acceptance Test Harness came from the community but [Sony Mobile] was the biggest contributor to actually getting traction on it"*. From Sony Mobile's perspective, it can contribute its internal acceptance tests to the community and have the community execute what Sony Mobile tests when setting up the next stable version. Consequently, it requires less work of Sony Mobile when it is time to test new stable version. From the community perspective I3 stated, *"an Acceptance Test Harness also helps the community and other companies to understand what problems that big companies have or small companies in terms of features or in terms other requirements on the system. So it's a tool where everyone helps each other"*.

5.5 Innovation outcomes

This theme is related to *RQ4* (see Table 3). In terms of measuring process and product innovation outcomes, Sony Mobile does not have any metrics. However, valuable insights were found during the interviews regarding what Sony Mobile has gained from the Jenkins and Gerrit community involvement. Following innovation outcomes are identified in the study.

1. Free features
2. Free maintenance
3. Freed up time
4. Knowledge retention
5. Flexibility
6. Increased turnaround speed

7. Increased quality assurance
8. Improved new product releases and upgrades
9. Inner source initiative

The most distinct innovation outcome is the notion of obtaining ***free features*** from the community, which have different facets [33, 132]. For projects maintained by Sony Mobile, such as the Gerrit Trigger plug-in, a noticeable amount of external contributions can be accounted for. Similarly, in communities where Sony Mobile is not a maintainer, they can still account for free work, but there requires a higher effort in lobbying and actively steering the community in order to maximize the benefits for the organization. Along also comes the ***free maintenance*** and quality assurance work, which renders better quality in the tools. Consequently, other than product innovations in tools as Jenkins and Gerrit, ***freed up time*** may be spent on other matters of high importance to Sony Mobile.

Correlated to the ***free work*** is the acknowledgement that the development team of six people in the Tools department will have a hard time keeping up with the external workforce, if they were to work in a closed environment. *"...I mean Gerrit has like lets say we have 50 active developers, it's hard for the tech company to compete with that kind of workforce and these developers at Gerrit are really really smart guys. Its hard to compete for commercial companies"*. Further on, *"...We are mature enough to know that we loose the competitive edge if we do not open up because we cannot keep up with hundreds of developers in the community that develops the same thing"*.

An organizational innovation outcome of opening up is the ***knowledge retention*** which comes from having a movable workforce. People in the community may move around geographically, socially and professionally but can still be part of the community and continue to contribute. I3, who took part in the initiation of many projects, recently left Sony Mobile but is still involved in development and reviewing code for his former colleagues which is in-line with the findings of the previous studies [101, 132]. In a another case, the knowledge tied to I3 would have risked being lost for Sony Mobile.

An outcome on software development in general after opening up is that when entering a new functional area or there is a need of a library, the mentality is to always look for open source first or co-develop. Developers try to make use of existing open source solutions to start with as an initial solution. The biggest challenge usually regards scaling the software to an enterprise level.

Before the paradigm shift and opening up of Sony Mobile, many tools were proprietary. Adapting these tools, such as the build server Electric commander, was cumbersome and it took long time before even a small fix would be implemented and delivered by the supplier. This created a stiffness whereas open source brought ***flexibility***. I2 quoted, *"...Say you just want a small fix, and you can fix that yourself very easily but putting a requirement on another company, I mean*

it can take years. Nothing says that they have to do it". This increase in the ***turnaround speed*** was besides the absence of license fees, a main argument in the discussions when looking at Jenkins as an alternative to Electric commander. As a result, the continuous integration tool chain could be tailored specifically to the needs of the product development team. I1 stated that *"...Jenkins and Gerrit have been set up for testers and developers in a way that they can have their own projects that build code and make changes. Developers can handle all those parts by themselves and get to know in less than 3 minutes whether or not their change had introduced any bugs or errors to the system"*. Ultimately, it provides ***quality assurance*** and performance gains by making the work flow easier for software developers and testers. Prior to the introduction of these tools there was one engineer who was managing the builds for all developers. In the current practice everybody is free to extend on what is given to them from tools department. It offers more scalability and flexibility [102].

I1 stated that besides the flexibility, the Tools department is currently able to make a *"...more stable tools environment in [Sony Mobile] and that sort of makes our customers of the tools department, the testers and the engineers, to have an environment that actually works and does not collapse while trying to use it"*. I2 mentioned that *"...I think its due to the part of open source and we are trying to embrace all these changes to our advantage. I think we can make high quality products in less time and in the end it lets us make better products. I think we never made as good product as we are doing today"*. Further exploration of this statement revealed the background context where Sony Mobile has ***improved*** in terms of handling all the ***new releases and upgrades*** in their phones compared to their competitors and part of its credit is given to the flexibility offered by the open source tools Jenkins and Gerrit.

The external knowledge obtained, in the form of different parts in the continuous integration tool chain, enabled better product development. However, the Tools department has to take the responsibility for the whole tool chain and not just its different parts, e.g. Jenkins and Gerrit, described by I5 as the next step in the maturity process. The tool chain has the potential to function as an enabler in other contexts as well, seeing Sony Mobile as a diversified company with multiple product branches. By opening up in the way that the Tools department has done, effects from the coupled OI processes with Jenkins and Gerrit may spread even further into other product branches, possibly rendering in further innovations on different abstraction levels [95]. A way of facilitating this spread is the creation of an ***inner source initiative*** which will allow for knowledge sharing across the different borders inside Sony Mobile, comparable to an internal OSS community, or as a bazaar inside a cathedral [138]. The tool chain is even seen as the foundation for a platform which is supposed to facilitate this sharing [94]. The Tools department is considered more mature in terms of contributing and controlling the OSS communities. Hence, the Tools department can be used as an example of how other parts of the organization could open up and work with OSS communities. I5

uses this when evangelizing and working on further opening up the organization at large, and describes how *"... They've been spearheading the culture of being active or in engaging something with communities"*.

6 Discussion

6.1 Opening up

The move to Android took Sony Mobile from a closed context to a neutral external arena for OI as described by Grotnes [55]. With this, the R&D was moved from a structured joint venture and an internal vertical hierarchy to an OI community. This novel way of using pooled R&D [143] can be further found on the operational level of the Tools department, which freely cooperates with both known and unknown partners in the Jenkins and Gerrit communities. From an OI perspective, these activities can be broken down into a number of outside-in and inside-out transactions. The Tools department's involvement in Jenkins and Gerrit and the associated contribution process are repetitive and bidirectional. Thus, this interaction can be classified as a coupled innovation process [50]. This also complies with Grotnes' description of how an open membership renders in a coupled process, as Jenkins and Gerrit communities both are free for anyone to join, compared to the Android platform and its Open Handset Alliance, which is invite-only [55].

The quantitative results provide further support for the hypothesis that both incumbents and small scale software-intensive organizations are involved in the development of Jenkins and Gerrit. This confirms that others than Sony Mobile also using these communities as external R&D resources and complimentary assets to internal R&D processes which is in line with the findings of existing OI literature [63, 131]. One possible motivation for start-ups or small scale organizations to utilize external R&D is their lack of in-house R&D capabilities. Incumbents exploit communities to influence not only the development direction, but also to gain a good reputation in the community as underlined by prior studies [34, 63].

Gaining a good reputation requires more than just being an active contributor. Stam [131] separates between technical (e.g. contributions and bug fixes) and social activities (e.g. organizing conferences and actively promoting the community), where the later is needed as complementary in order to maximize the benefits gained from the former. Sony Mobile and the Tools department have evolved in this vein as they are continuously present at conferences, hackathons and in online discussions. Focused on technical activities, the Tools department have progressively moved from making small to more substantial contributions. Along with the growth of contributions, they have also matured in their contributions strategy. As described in Section 5.2, the intent was originally to keep the Gerrit Trigger plug-in enclosed. This form of selective revealing [62] has however been minimized due to a more opened mindset. As a consequence of the openness more plug-ins were initiated and development time reduced.

Although the adoption of Jenkins and Gerrit came along with an adaption to the Android development, it was also driven bottom-up by the engineers since they felt the need for easing off the complex integration tool chain and building process as mentioned by Wnuk et al. [148]. As described in Section 5.1, this process was not free of hurdles, one being the cultural and managerial aspect of giving away internally developed intellectual property [70]. The fear to reveal intellectual property was removed thanks to the introduction of an OSS review board. Having an internal champion to give leverage to the needed organizational and process changes, convince skeptical managers [63], and evangelize about open source was a great success factor, also identified in the inner source literature [97].

6.2 Determinants of openness

When discussing if something should be made open or contributed, an initial distinction within the Tools department regarding the possible four cases is made:

1. New projects created internally (e.g. Gerrit Trigger)
2. New features to non-maintained projects (e.g. Gerrit)
3. External feature requirement requests to maintained projects (e.g. Gerrit Trigger)
4. External bug reports to maintained projects (e.g. Gerrit Trigger).

The first two may be seen as an inside-out transaction, whilst the two latter are of an outside-in character. All have their distinct considerations, but one they have in common, as described in Section 5.2, is whether Sony Mobile will benefit from it or not. Even though the transaction cost is relative low, it still needs to be prioritized against the current needs. In the case of the two former, if a feature is too specific for Sony Mobile's case it will not gain any traction, and it will be a lost opportunity cost [92].

The fact that Sony Mobile considers their supportive tools, e.g. Jenkins and Gerrit, as a non-competitive advantage is interesting as they constitute an essential part of their continuous integration process, and hence the development process. As stated in regards to the initial intent to keep Gerrit Trigger internally, they saw a greater benefit in releasing it to the OSS community and having others adopt it than keeping it closed. The fear that the community was moving in another direction, rendering in a costly need of patch-sets and possible risk of an internal fork, was one reason for giving the plug-in to the community [134]. Wnuk et al. [148] reasons in a similar manner in their study where they differentiate between contributing early or late to the community in regards to specific features. By going with the former strategy, one may risk losing the competitive edge, however the latter creates poteniaally high maintenance costs.

Sony Mobile is aware of the fact that increased mobility [22] poses a threat to the Tools department as it is not possible for them to work in the OSS communities' pace due to the limited amount of resources [22]. Consequently, it may end up damaging the originally perceived competitive advantage by lagging behind. On the other hand, openness gives Sony Mobile an opportunity to have an access to pragmatic software development workforce and also, Sony Mobile does not have to compete against the community. Additionally, by adopting a first mover strategy [93] Sony Mobile can use their contributions to steer and influence the direction of the community.

6.3 Requirements engineering and OI

The Tools department may be viewed as both a developer and an end-user, making up a source of requirements as can often be seen in Open Source Software Development (OSSD) [124]. This applies both internally (as a supplier and an administrator of the tools), and externally (as a member of the communities). From the RE perspective, they are their own stakeholder, competing with other stakeholders (members) in ecosystems, e.g. Jenkins and Gerrit communities, e.g. in the elicitation and prioritization processes. In regards to elicitation, the level of influence the Tools department has may allow them to affect what sources requirements are taken from, e.g. themselves, but also from those with similar intents with regards to features and the direction of the community. Same logic applies to prioritization, as the level of influence affects how the Tools department gets their plans realized compared to other stakeholders.

The power structure varies as their roles shift between being a maintainer and an ordinary member with different levels of influence. A related concern suggested by Alspaugh et al. [8] is that stakeholders who are not developers themselves are often neither identified nor considered. This may lead to that certain areas are forgotten or neglected which is in contrast to Wnuk et al. [148] who state that adoption of OI makes identifying stakeholder's needs more manageable. Further, this brings an interesting contrast to traditional RE where non-technical stakeholders often need considerable help in expressing themselves. RE in OI applied through OSS hence can be seen as quicker, light-weight and more technically oriented than traditional RE [125].

For the Tools department, scalability was an overseen functional area in the sense that both Jenkins and Gerrit had relatively weak support for large production environments as was needed by Sony Mobile. No other stakeholder with a similar environment size had stepped forward earlier. An interesting note in this case is that even though the two tools were lacking functionality, Sony Mobile still proceeded with them as issues could be fixed over time. In OSSD, one often needs to have a high authority level or have a group of stakeholders backing up the intent. Sony Mobile has been very successful in this aspect due to the Tools department involvement inside these communities [34]. This results in Sony Mobile

employees being usually high up in the governance organization due to their high commitment and good track record. The Tools department combines their position in the communities together with openness in terms of helping competitors and interacting in social activities [131] (e.g. developers conferences [80]) in order to attract quiet stakeholders, both in terms of influencing the community [33], but also to get access to others knowledge which could be relevant for Sony Mobile. Communication in this requirements value chain [47] between the different stakeholders, as well as with grouping, can be deemed very ad-hoc as OSS RE is in general [125]. This correlates to the power structure and how influence may float between different stakeholders.

Along with the intent to influence the community, social interaction between the stakeholders is stressed as an important aspect to resolve conflicts and to coordinate dependencies in distributed software development projects [109]. The Tools department preference for live meetings over the otherwise available electronic options such as mailing lists, issue trackers and discussion boards, is due to time differences and lag in discussions that complicate implementation of larger features. Open source hackathons [126] is the preferable choice as it brings the core stakeholders together which allows for informal negotiations [47] and a live just-in-time requirements process [43], meaning that requirements are captured in a less formal matter and first fully elaborated during implementation. As highlighted in section 5.3, feature-by-feature collaborations is also a common practice. This is also due to the ease in communication as it may be performed between two single parties. Hence, it may be concluded that communication in this type of distributed development is a critical challenge, and in this case overcome by live meetings and keeping the number of collaborators per feature low.

Another interesting reflection on the feature-by-feature collaborations is that these may be performed with different stakeholders, i.e. relations between stakeholders fluctuate depending on their respective interests. This objective and short-term way of looking at collaborations imply a need of standardized practices in a community for it to be effective. Furthermore, it highlights the need to continuously analyze the present stakeholders in the community in order to identify those with similar intents, both for possible collaborations and to find partners in order to gain leverage in prioritization processes, for example.

Knauss et al. [80] highlighted the differences between two levels through which requirements are communicated between stakeholders in an open-commercial software ecosystem, namely a strategic and an emergent requirements flow. These may also be transferred to the Jenkins and Gerrit communities. In the emergent flow, end-users and developers continuously find new ways of using the software and present their solutions for discussion on work-item level in the open communication channels available, i.e. this is similar to post-hoc assertions [8] through prototypes and plug-ins, and communication practices that are generally performed in OSS communities, including those surrounding Jenkins and Gerrit. The strategic flow regards how business goals and strategies originating from different stake-

holders are aligned and considered in an ecosystem. From Sony Mobile's perspective, improved scaling possibilities of the production environments can be regarded as a business goal which needed to be echoed and rallied for in the OSS communities for traction to be gained. This separation between an emergent and a strategic flow may implicate the need for different practices. In the example where Sony Mobile was in need of a more scalable production environment, a high degree of social pressure and traction-creation was needed to steer the community and to negotiate with other stakeholders. Features that are more narrow and low-level may be implemented on post-hoc assertions and communicated with much less effort through e.g. mailing lists or issue trackers [125]. This is an area that needs further investigation in future research, especially how it relates to the different sub-practices of RE, e.g. elicitation and prioritization.

The way in how requirements are elicited, analyzed, specified and stored in the Jenkins and Gerrit communities include multiple instances, such as issue trackers, email-lists and bulletin board-discussions. Scacchi [125] labels these ways of communicating and describing features as software "informalisms", where artifacts which capture the requirements are exposed to the community, to read, understand and discuss their implementation proposals. As identified in Section 5.3 a preferable way to present and sell such informalisms, i.e. feature suggestions, is with prototyping as is commonly used in OSSD [8, 53], alongside referring to the attributes in competing products or previous versions of a project. This way of presenting requirements combines well with how the developers in the Jenkins and Gerrit communities tend to present ready made solutions rather than specifying the problem a priori.

With the highlighted use of live-meetings and social events to communicate and discuss requirements, it is important that these are recorded as other informalisms so that all stakeholders and members of the community gets to know about what, how and why some things were implemented or decided. This implies a higher importance of being socially present in a community other than just online if a stakeholder wants to stay aware of important decisions and implementations. Another reason would be so that the stakeholder may maintain or grow its influence and position in the governance ladder. Hence, firms might need to revise their community involvement strategy and value what their intents are in contrast to if an online presence is enough.

As the requirements are distributed among multiple "storages" in the Jenkins and Gerrit communities, it is hard to maintain quality factors such as consistency, completeness, correctness and traceability as in traditional RE (e.g. [88]). Even though the Jenkins and Gerrit communities lack these features, the two tools are still entrusted to act as critical parts of Sony Mobiles' development process which supports the general trend that OSSD can generate high quality applications even for business critical applications [8].

6.4 Testing process and OI

In both Jenkins and Gerrit the focus has in general been on manual test cases. However, the maturity process has begun where the communities are moving more towards automated testing, with the Jenkins community leading. The openness of the Tools department led them to participate in the testing part of Jenkins community and to use its influence to rally the traction towards it amongst the other stakeholders in the community. This is especially important for the Jenkins community due to the rich number of settings offered by the plugins.

Currently, the Gerrit community is following the Jenkins' community patch, as stressed by I2. With this move towards automated testing, quality assurance will hopefully become better and enable more stable releases. These are important aspects and business drivers for the Tools department as Jenkins and Gerrit constitute the critical parts in Sony Mobile's continuous integration tool chain. Seeing from this perspective, a trend may be visualized in how the different communities are becoming more professionalized in the sense that the tools make up business critical assets for many of the stakeholders in the communities, which motivates a continuous effort in risk-reduction [62, 105].

The move towards automated testing also allowed for the Tools department to contribute their internal test cases. This may be viewed as profitable from two angles. First, it reduces work load internally, and second, it secures that settings and cases specific for Sony Mobile are addressed and cared for. The test cases may to some extent be viewed as a set of informal requirements, which secure quality aspects in regards to scalability for example which is important for Sony Mobile [13]. Similar practices, but much more formal, are commonly used in more traditional (closed) software development environments. From a community perspective, other stakeholders benefit from this as they get the view and settings from a large environment which enable them to grow as well.

As can be noted in Fig. 5, the focus is on forward and re-engineering. An interested concern is when and how much one should contribute in regards to bug fixes and what should be left for the community, because some bug fixes are very specific to Sony Mobile and the community will not gain anything from it. As discussed earlier, Sony Mobile has the strategy of focusing on issues which are self-beneficiary. Although, to be able to keep the influence and strategic position in the communities, the work still has to be done in this area as well.

6.5 Innovation outcomes

The focal point of the OI theory is value creation and capture [24]. In the studied case, the value is created and captured through their involvement in the Jenkins and Gerrit communities. However, measuring that value using key performance indicators is a daunting challenge. Edison et al. [40] confirm a limited number of measurement models, and that the existing ones neither model all innovation aspects, nor say what metric can be used to measure a certain aspect. Furthermore,

existing literature is scarce in regards to how data should be gathered and used for the metrics proposed by literature. As expected, interviewees mentioned that Sony does not have established mechanisms in place to measure their performance before and after the Jenkins and Gerrit introduction. This confirms the findings of Edison et al. [40]. However, from the qualitative data collected from the interviews we specifically looked for two types of innovations, product innovations in the tools Jenkins and Gerrit, and process innovation in Sony Mobile's product development. Other types, specifically market and organizational innovation were considered but not identified. Further, we did not diffrentiate between different impacts of innovation, such as incremental, really new or radical [48].

By taking an active part in the knowledge sharing and exchange process with communities [33, 132], the Tools department enjoys the benefits of contributions extending the functionality of their continuous integration tools. Another benefit is the free maintenance and bug corrections and the test cases extension for further quality assurance. These software improvements and extensions can to a varying extent to labeled product innovations depending on what definition to be used [40]. This may also be viewed from the process innovation perspective [4] as Sony Mobile gets access to extra work force and a broad variety of competencies, which are internally unavailable [33]. The interviewees admit to that even a large scale software-intensive organization cannot keep up the technical work force beyond the organization's borders and there is a huge risk of losing the competitive edge by not being open. This is an acknowledgement to Joys law [86] *"No matter who you are, most of the smartest people work for someone else"*. Hence, it is vital to reach work force beyond organizations boundaries when innovating [24]. And as earlier described, knowledge is still retained as people move around inside the community.

Furthermore, these software improvements and product innovations affect the performance and quality of the continuous integration process used by Sony Mobile's product development. Continuous integration as an agile practice [11] enables early identification of integration issues as well as increases the developers' productivity and release frequency [130]. With this reasoning, as reported elsewhere [95], we deem that the product innovations captured in Jenkins and Gerrit transfer on as process innovation to Sony Mobile's product development. The main reason behind this connection is viewed as the possibility to tailoring and flexibility that the OSSD permits. By adapting the tool chain to the specific needs of the product development the mentioned benefits (e.g. increased build quality and performance) are achieved and waste is reduced in the form of freed up hours, which product developers and testers may spend on alternative tasks as confirmed by Moller [100]. The business impact in the end is a reduced time to market and increased quality of products. This latter statement is however not confirmed as metrics are not available and is out of scope of this paper.

Another process innovation, which could also be classified as an organizational innovation outcome [4], is the inner source initiative where Sony Mobile wants to

spread the tool chain, as well as to build a platform (i.e. software forge [94]) for sharing built on the tool, intra-organizationally to other business units within Sony. This may be seen as an intra-organizational level OI as described by Morgan et al. [101]. By integrating the knowledge from other domains, as well as opening up for development and contributions, this allows a broader adoption and a higher innovation outcome for Sony Mobile and neighboring business units, as well as for communities. Organizational change in regards to processes and structures and related governance issues, would however be one of many challenges needed to address [101]. Seeing Sony Mobile as a multinational corporation with a wide spread of internal culture this could prove to be a difficult task.

6.6 Openness of Tools software vs. Product software

An important finding of this study related to openness is that Sony Mobile only opens up its non-competitive tools that are not the part of the revenue stream. I3 stated that *"... Sony Mobile has learnt that even collaborating with its worst competitors does not take away their competitive advantage, rather they bring help for Sony Mobile and becomes better and better"*. Consequently, Sony Mobile's internal development environment has become more stable and gained access to the skilled workers from the community that would otherwise have come with significant costs. As a result, the communities receive contributions from a large scale software-intensive organization. This raises a discussion point of why Sony Mobile limits its openness to non competitive tools, despite knowing that opening up creates a win-win situation for all stakeholders involved. Furthermore, it remains an open question why the research activity related to OI in SE is low as confirmed by the results of a mapping performed on the area [107].

In the light of the mapping study conducted prior to this study, it would be fair to state that the SE literature lacks studies on OI using software both with a direct and indirect competitive advantage. Organizations have a tendency to open proprietary products when they loose their value, and spinning off is a one way of re-capturing the value by creating a community around it [96]. This implication paves way for future studies using proprietary solutions as units of analysis. Moreover, it will lead to contextualization of OI practices, which may or may not work under different circumstances. Therefore, the findings could also be used to strengthen the OI theory weakness mentioned by Mowery [104] that the OI phenomenon lacks contextualization. It is also important to note that this study focuses on OI via OSS participation, which is significantly different from the situation where OI is based on open source code for the product itself (like Android or Linux). The implication of this difference is that we need to explore that situation to see if there are other patterns in these OI processes.

7 Conclusion

This study has focused on OI in SE at two levels: 1) innovation incorporated into Jenkins and Gerrit as software products, and 2) how these software improvements affect and innovate the continuous integration process used by Sony Mobile's. By keeping the development of the tools open, the in- and out-flows of knowledge between the Tools department and the OSS communities bring improvement to Sony Mobile and innovate the way how products are developed. This type of openness should be separated from the cases where OSS is used as a basis for the companies' product or service offering, e.g. as a platform, component or full product [134]. To the best of our knowledge, no study has yet focused on the former version, which highlights the contribution of this study and the need for future research of the area.

Our findings suggest that both incumbents and many small scale organizations are involved in the development of Jenkins and Gerrit (**RQ1**). Sony Mobile may be considered as one of the top contributors in the development of the two tools. The main trigger behind adopting OI turned out to be a paradigm shift, moving to an open source product platform **(RQ2)**. Sony Mobile's opening up process is limited to the tools that are non-competitive and non-pecuniary. Furthermore, for Sony Mobile to open up a project or e.g. to put effort into a feature implementation, there needs to be a clear incentive or profit motivation **(RQ3)**.

In relation to the main innovation outcomes from OI participation (**RQ4**), we discovered that Sony Mobile lacks key performance indicators to measure its innovative capacity before and after the introduction of OSS at the Tools department. However, the qualitative findings suggest that it has made the development environment more stable and flexible. One key reason, other than contributions from communities, regards the possiblity of tailoring the tools to internal needs. Still, it is left for future research efforts to further investigate in how OI adoption affects product quality and time to market.

When looking at the impact of OI adoption on requirements and testing processes (**RQ5**), Sony Mobile uses dedicated resources on the inside to gain influence, which together with an openness toward direct competitors and communities is used to draw attention to issues relevant for Sony Mobile, e.g. scalability of tools to large production environments. Social presence outside of online channels is highly valued in order to manage communication challenges related to distributed development. Another way of tackling such challenges regards co-creation on a feature-by-feature basis between two single parties. Choice of partner fluctuates and depends on feature in question and individual needs of the respective parties. Further on, prioritization is made in regards to how an issue or feature may be seen as beneficial, in contrast towards the pressing needs of the moment. Regarding testing, much focus is directed towards automating test activities in order to raise quality standards and professionalize communities to company standards.

Findings of the study are limited to software-intensive organizations with the

similar context, domain and size as Sony Mobile. It is also worth mentioning that stakeholders involvement in Jenkins and Gerrit suggest that their continuous integration process is comparable to Sony Mobile thus, we believe that findings of this study may also be applicable to incumbents as well as small software-intensive organizations identified in this study (see Fig. 4).

For future work, it would be interesting to investigate whether or not OI depends on the contextual factors such as domain, size and experience etc. Moreover, one of the key findings of the study is that the testing process in OI is very much a work in progress shifting from manual test cases to automated test cases in order to test Jenkins plug-ins using an acceptance test harness. Therefore, An exploratory study with the intention of understanding the acceptance test harness could be a good candidate for the future work.

Regarding future work for RE in OI, it would be interesting to investigate how RE practices (e.g. elicitation and prioritization) towards OSS communities differ on a strategic and more practical lower level, and how these should be constructed to optimise the innovation outcome and bridge with a firms internal RE process. Also, account should be taken to the identified challenges of communication and fluctuating short-term partnerships, and how this may demand a continuous analysis of present and up-coming stakeholders. Another interesting topic for future work includes investigation of how stakeholders in a community should manage an increased need for a social presence outside the traditional online communication channels.

CHAPTER IV

SOFTWARE TESTING IN OPEN INNOVATION: AN EXPLORATORY CASE STUDY OF THE ACCEPTANCE TEST HARNESS FOR JENKINS

Abstract

Open Innovation (OI) has gained significant attention since the term was introduced in 2003. However, little is known whether general software testing processes are well suited for OI. An exploratory case study on the Acceptance Test Harness (ATH) is conducted to investigate OI testing activities of Jenkins. As far as the research methodology is concerned, we extracted the change log data of ATH followed by five interviews with key contributors in the development of ATH. The findings of the study are threefold. First, it highlights the key stakeholders involved in the development of ATH. Second, the study compares the ATH testing activities with ISO/IEC/IEEE testing process and presents a tailored process for software testing in OI. Finally, the study underlines some key challenges that software intensive organizations face while working with the testing in OI.

1 Introduction

Open Innovation (OI) is an emerging paradigm in Software Engineering (SE) which lacks empirical evidence for software-intensive organizations. In 2003, Chesbrough defined OI as follows [24]: *"A paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external path to markets, as they look to advance their technology"*. One of the most well

known ways for enabling OI in the software-intensive organizations is the use of Open Source Software (OSS). However, it is important to acknowledge that OI and OSS are *not* the same. In order for OSS to be used as an example of OI, firms' OSS activities must be in line with their business model to create and capture value. OI is more transactional in nature, compared to OSS, where firms try to leverage external knowledge to accelerate their internal innovation process and in return, contribute back to the community by adopting a selective revealing strategy [62].

Prior to this study, we conducted a systematic mapping study [107] on OI in SE to identify the research in the field. The study shows that empirical studies on the role of testing in OI are scarce. Furthermore, software testing in OI entails a dual role: 1) to verify the functions and characteristics of open components and services, supplied by the community, and 2) to verify the functions and characteristics of services delivered to stakeholder higher up in the value chain (e.g. internal customers, software developers and testers). Furthermore, it is still unknown whether or not the general practices of software testing are feasible to deal with the challenges of OI.

This paves way for this exploratory case study with the main focus of software testing in OI. The object of study is Jenkins, an open source build server [2]. The main objective of this study is to identify the top contributors to the Acceptance Test Harness (ATH) which is part of the Jenkins project (see Section 3.1) and explore the testing processes used to test Jenkins, using ATH. Furthermore, this paper presents the key challenges faced by one of the key contributors to Jenkins (see Section 3.3).

2 Research Design

In order to explore software testing activities in OI, we launched a case study [122] of an OSS project, studied as an instance of OI. The focus of the study is on the initiation and development of the ATH to test Jenkins in an automated fashion. We conducted the following steps: first we mined the ATH source code repository and extracted the change log data, using CVSAnly, to characterize the top ATH contributors in the Jenkins community. Then, face to face semi-structured interviews were conducted with the key software developers of ATH (see Table 1). Thirdly, we analyzed the OI testing by relating the process to a general test process, and identified key challenges for OI testing.

2.1 Research Questions

Our general interest in understanding OI aspects of software testing are detailed in three research questions:

RQ1: Who are the top stakeholders involved in the development of ATH and are those stakeholders the same as the contributors of Jenkins?

RQ2: Do ATH testing activities adheres to ISO/IEC/IEEE 29119 testing standard?

RQ3: What are the key challenges associated with testing in OI?

As a point of reference for general test processes, we used the ISO/IEC/IEEE 29119 testing standard to compare the ATH testing process with *(RQ2)*. This lead to identifying the key contributors *(RQ1)* in ATH, followed by interviews with the developers of ATH to find out the possible hurdles *(RQ3)* in the process. To be more specific, *RQ1* is formulated to investigate the homogeneity of the community, to see whether or not the top contributors of ATH are the same as for the core of Jenkins. *RQ1* is answered by mining the change log data from the online ATH GitHub repository. It is to be noted that the change log mining is also used to identify the key contributors of ATH for interviews. *RQ2* aims at exploring the adherence of ATH to ISO/IEC/IEEE 29119 testing standard. In order to analyze the OI test process, the ISO/IEC/IEEE standard is a reference model for traditional testing. Abdou et al. [5] used ISO/IEC/IEEE standard to compare it with the OSS testing process. However, we followed ISO/IEC/IEEE standard to compare it with the OI testing process. There are number of standards available for testing processes, such as ISO/IEC TR 19759, BS 7925-2 and ISO/IEC WD 29119-2 [1,5,17,118], while we used the latter since is is quite recent and internationally accepted as as standard. ISO/IEC/IEEE 29119 supports various software development life cycles including spiral, waterfall and agile models. Unlike all previously mentioned models, it covers non functional testing, risk based testing and static testing. The change log data extracted to answer *RQ1* helped us identifying the key interviewees in order to explore the challenges associated with ATH. *RQ3* is used to explore the key challenges associated with OI in SE. *RQ2* and *RQ3* are answered using interviews with the key contributors from the Jenkins community.

2.2 Case Selection and Unit of Analysis

Jenkins is the leading open source continuous integration server [2]. It offers more than 1000 plugins to support building and testing virtually any project built in Java. These tests can be also run with specific version of the Jenkins core and a combination of plugins. Over the passage of time, the number of test cases for Jenkins has steadily increased to over 300. In order to test Jenkins with automated tests, one of the key contributors (referring to an anonymous company's employee) initiated the ATH together with the community. ATH consists of a reusable harness that can be used by plugin developers and users to write functional test cases. These test cases can be run against Jenkins plugins that are deployed in different configurations. The focus of this case study is to explore OI activities in ATH from the companies' perspective rather than community's perspective.

Table 1: Interviewees description

Anonymous name	Involvement	Experience	Role
Interviewee 1	Jenkins and ATH	8 Years	Tools manager for Jenkins and contributing to ATH
Interviewee 2	Jenkins and and ATH	6 Years	Team lead and ATH contributor
Interviewee 3	Jenkins	7 Years	Former tools manager Jenkins
Interviewee 4	Responsible for Jenkins and Gerrit build artifacts and channel distribution	8 years	Software Architect
Interviewee 5	OSS policy maker	More than 20 Years	Manager responsible for overall OSS strategy

3 Results and Analysis

This section presents the findings from the change log data analysis and the semi-structured interviews with the contributors of ATH.

3.1 Stakeholders in ATH

The associations of key contributors were identified using their email addresses followed by checking their public profiles on GitHub. In Fig. 1, Cloudbees and Redhat are the biggest contributors to the development of ATH. Surprisingly, the third biggest contributor turned out to be Munich University which is an indication of strong ties between the Jenkins community and academia [22]. It is to be noted that the biggest contributor in Jenkins is *not* among the top for ATH. However, we chose to interview key contributors to Jenkins, since they initiated the idea of ATH and convinced the community to start testing Jenkins in an automated fashion.

Interview data suggests that the testing process of Jenkins is the least attended process since the community was using an old manual approach for the testing of Jenkins. Openness lead them to consider switching from the manual to an automated testing process. Initially, the idea of testing Jenkins in an automated fashion came from the community. An interviewee 3 stated, *"... the idea of acceptance test harness came from the community but [our company] was the biggest contributor to actually getting traction on it"*.

Selenium is a test harness that tests Jenkins from outside, using automated tests. Originally it was written in Ruby, while Jenkins generally have unit test cases written in HTML that only test a particular plug-in in a unit test manner. In 2011, this became a significant problem for two reasons. Firstly, the HTML solution did not scale up to a large number of plug-ins, and secondly, the community primarily used Java, and thus the Ruby implementation became a bottleneck regarding competence.

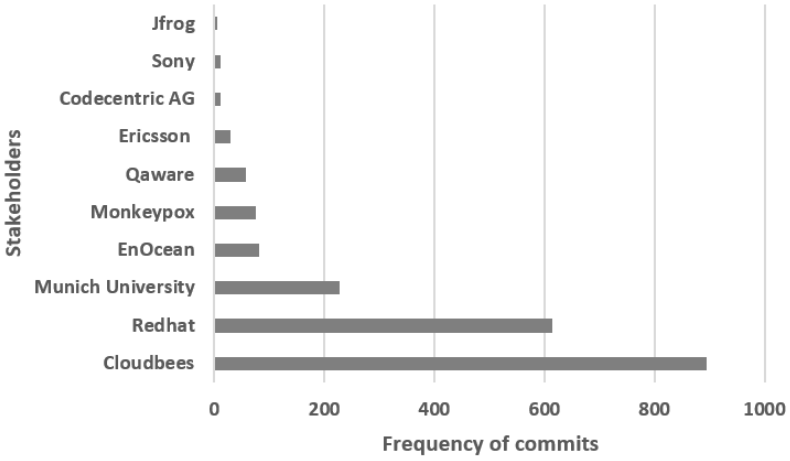


Figure 1: Key contributors to the Acceptance Test Harness.

Therefore, the main contributor took on the scaleability issue and together with the Jenkins community decided to rework the test harness into Java. The ATH is comprised of a reusable harness that can be used by users and plug-in developers to write functional test cases. An interviewee 3 stated that, *"...from [the company's] perspective, we can contribute our internal acceptance test cases to the community and have the community actually to execute those tests when its time to test a new stable version and upgrading. Similarly, it helps other companies and the community to do the same. Therefore, it's a tool that helps everyone"*.

The programming language (Ruby) and scaleability issues of the Selenium(HTML) test harness came up as a bottleneck to adopt it for the Jenkins testing, since the Jenkins community works with Java. Therefore, the ATH is created to port selenium test cases into Java unit test cases in order to test the Jenkins through automated acceptance test cases.

3.2 The ATH Testing Process

In this section we compare the ATH testing process to the ISO/IEC/IEEE 29119 testing standard. The structure below follow the main phases of the standard.

Test Design and Implementation

The OI testing is less formalized than described in the standard. ATH does not have an explicit test plan and there is no formal identification of risks (see Fig. 2). The features to be tested are prioritized according to the community's or contributor's

Table 2: Differences between ISO/IEC/IEEE 29119 and the OI testing process

Phases	ISO/IEC/IEEE Test Process	Acceptance test harness framework
Test Design and Implementation	• Identify Feature Sets • Derive Test Conditions • Derive Test Coverage Items • Derive Test Cases • Assemble Test Sets • Derive Test Procedures	• Identify Independently • Derive Testable Aspect(s) • Derive Test Cases • Documentation
Test Environment Set-Up and Maintenance	• Establish Test Environment • Maintain Test Environment	• Select Test Environment
Test Execution	• Execute Test Procedure(s) • Compare Test Results • Record Test Execution	• Test thoroughness depends on developers expertise • Execute • Submit
Test Incident Reporting	• Analyze Test Result(s) • Create Incident Report	• Review or Vote • Accept • Console reports • Textfile reports

internal needs *independently* without consulting other stakeholders in the project. *Testable objects* (plugins) are based on the specific interests from volunteers who to choose take over the testing for a component (see Table 2).

Test Environment Set-Up and Maintenance

According to the ISO/IEC standard, test cases are derived by determining the preconditions, post-conditions, input values and expected outcomes. However, in the absence of a test plan, ATH does not have test completion criteria that measure the test coverage of the Jenkins plugins. An interviewee 1 stated that, *"... the existence of acceptance test cases does not guarantee that Jenkins would not blow up every time we upgrade the core or its plugins since it is so volatile with with different plugins and configurations. However, it can give us a fair indication what went wrong and enable to fix the problem.'*. In most OSS projects, testers move directly to test execution from test design and implementation in order to execute test procedures, since there are no specific requirements for the test environment in ATH (see Fig. 2). However, ATH has an abstraction called JenkinsController

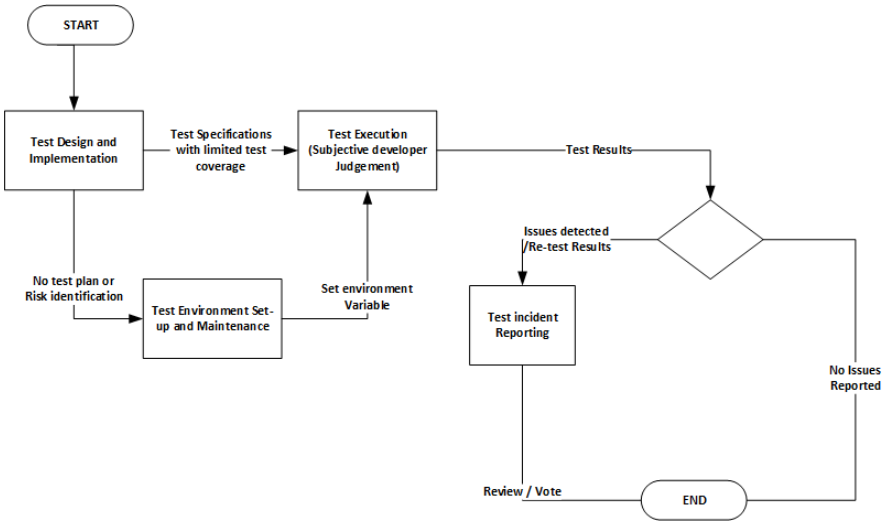


Figure 2: The ATH testing process

that allows using different logic for starting/stopping Jenkins. It is used to run the same set of tests against many different ways of launching Jenkins, such as through Java, JBoss, a Debian package, etc. To select a controller, run the test with the *TYPE environment* variable set to the controller ID. Common configuration of controllers can be done through environment variables, and the following controllers are available in ATH.

- Winstone controller (TYPE=winstone)
- Winstone Docker controller (TYPE=winstone_docker)
- Existing Jenkins' controller (TYPE=existing)
- Tomcat controller (TYPE=tomcat)
- JBoss controller (TYPE=jboss)
- Ubuntu controller (TYPE=ubuntu)
- CentOS controller (TYPE=centos)
- OpenSUSE controller (TYPE=opensuse)

Due to the open plug-in nature of the Jenkins it is difficult to have a one controller for all the above mentioned configurations. Therefore, the ATH provides different types of controllers to run the acceptance test suite against many different environments by setting the environment variable.

Test Execution

The Test Execution process generally begins with a developer, testing the local copy of Jenkins after checkout and the thoroughness of test cases depends on the expertise and judgment of the developer (see Fig. 2). However, in order to ensure the quality of acceptance test cases, a video tutorial is available on the Jenkins web page that shows how to write an acceptance test case. ATH uses WebDriver to execute tests, and the developers have the option to choose the browser by using the BROWSER environment variable. The following browser variables are compatible with ATH to execute the test suite.

- firefox (default)
- ie
- chrome
- safari
- htmlunit
- phantomjs

Test Incident Reporting

Test incident reporting in ATH depends on the discussions in the mailing lists, which solicit feedback from interested stakeholders, and the voting system. ATH allows *Console* and *Textfile* reports. The *Console Reporter* logs the plugin and its version to standard output. The *Textfile Reporter* creates a properties file in the target folder containing a list of plugin names and their versions, prefixed by the test name. It can be very useful when users want to be able to see which plugins and their versions that were tested with a particular version of Jenkins Core. Note that the reporting can be performed by any stakeholder involved in the development of ATH. However, the community uses the voting system to prioritize the most critical bugs (see Table 2). Moreover, test results are categorized into three categories: 1) OK, 2) Warn, and 3) Fail. Test results are considered OK if the tests passed without any problems. Warning indicates a functionality malfunction, but it does not cause the whole Jenkins to fail. On the contrary, failure indicates a fatal error of the whole Jenkins or a critical issue that affect all jobs, i.e. developers have not been to start Jenkins or save the configuration of any job.

Interviewees stated that external bugs are reported through the JIRA tool and internal bugs are reported using emails. The prioritization is based on the most pressing needs of software developers at the contributor. Additionally, all the test cases are written in Java using JUnit. An interviewee 3 stated, *"... if it is sort of a make or break we fix it ourselves and then make a pull request and if its not, we report it. May be someone from the community in the future will fix it"*. Other

stakeholders in the ATH also do the same, by taking the most important issues according to their needs and fix them first. However, it is important to show to the community that we want to contribute to the project as a whole and not just our part, as mentioned by Dahlander [34]. An interviewee 3 stated *"... sometimes, if we have a big issue someone else may have it too and we can focus on fixing other bugs so, we try to forward as many issues as possible."*. This finding is inline with the findings of Henkel et al. [63].

3.3 Software Testing Challenges in OI

One of the biggest testing challenges is to have a complete coverage as there are many different configurations and setups available due to the open plug-in nature of Jenkins. As one interviewee 1 stated *"... Jenkins is composed of so many small plugins thereby, every time [company] upgrade core and plugins for master services it blows up"*. Therefore, running the ATH test suites before applying an upgrade gives the team an indication about what might fail during the process. In addition to this quote, the interviewee further elaborated that they have some acceptance tests and the test suite is getting bigger, and therefore urges for a need to set up an automatic building (ATH) for every patch set. The second major challenge is related to the availability of resources, as an interviewee 3 stated *"... the hot shots in the community are really busy and do not have enough time to take on some of the most daring challenges we face. Therefore, sometimes it is frustrating to get an answer from the community quickly."* as confirmed by Morgan et al. [102]. This also traces back to our comparison (see Table 2) where stakeholders prioritize and fix their issues independently, without taking other stakeholders into account.

4 Conclusions

This case study explored testing activities in Jenkins, using ATH and compared the testing activities of ATH with the ISO/IEC/IEEE testing standard. We extracted the change log data of ATH in order to identify the major contributors (**RQ1**). In the interviews, we found out that although the initial idea of ATH came from the community, the major Jenkins contributor brought ATH to the community's attention at hackathons. Additionally, along with Cloudbees and Redhat, Munich university came out as a third biggest contributor, which suggests strong ties between the Jenkins community and industry. Further, we compare the difference between the ATH testing process and the ISO/IEC/IEEE testing standard (see Table 2). The key difference is that the ATH does not have a test plan and therefore, there is no test completion criteria that measures the test coverage of all Jenkins plugins. The thoroughness of the acceptance test cases depends on the subjective judgment of developers (**RQ2**). Finally, we identified key challenges for the testing process in

OI (**RQ3**). For example, due to the huge number (1000+) of open Jenkins plugins and its nature with many settings, it is a daunting task to have a complete test coverage. However, the ATH makes the whole Jenkins defect detection process better by pointing to defects in faulty plugins and thereby, enables developers to take corrective actions more efficiently.

It can be concluded that the ATH testing process does not strictly adhere to the ISO/IEC/IEEE testing standard because testable features are identified by software engineers independently without any formal test plan. The test coverage is dependent on software engineers subjective judgment and hence, it is very difficult to achieve the complete test coverage. Furthermore, different controllers to run acceptance test cases against many different environments is an indication that the standard Software Engineering testing process needs to be adapted to deal with challenges of OI. Finally, it is worth mentioning that Jenkins is a development infrastructure and ATH is used to test this infrastructure. In effect, ATH may not necessarily a representative of regular software example.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] ISO/IEC/IEEE 29119 Software Testing Standard.

[2] The jenkins gerrit trigger plugin open source project on ohloh. Accessed: 2014-07-08.

[3] Source - gerrit - link to the source browser. - gerrit code review - google project hosting. Accessed: 2014-06-24.

[4] *Oslo Manual – Guidelines for collecting and interpreting innovation data*. OECD and Eurostat, 3rd edition, 2005.

[5] T. Abdou, P. Grogono, and P. Kamthan. A conceptual framework for open source software test process. In *36th Annual Computer Software and Applications Conference Workshops (COMPSACW)*, pages 458–463, July 2012.

[6] Oliver Alexy, Joachim Henkel, and Martin W. Wallin. From closed to open: Job role changes, individual predispositions, and the adoption of commercial open source software development. *Research Policy*, 42(8):1325 – 1340, 2013.

[7] Robert C. Allen. Collective invention. *Journal of Economic Behaviour and Organization*, 4(1):1 – 24, 1983.

[8] Thomas A Alspaugh and Walt Scacchi. Ongoing software development without classical requirements. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 165–174. IEEE, 2013.

[9] K. Balka, C. Raasch, and C. Herstatt. How open is open source? - software and beyond. *Creativity and Innovation Management*, 19(3):248–56, 2010.

[10] M. Bano. Aligning services and requirements with user feedback. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 473–478, Aug 2014.

[11] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. *The agile manifesto*. 2001.

[12] Fiona Beyer and Kath Wright. Comprehensive searching for systematic reviews: a comparison of database performance. 2011.

[13] Elizabeth Bjarnason, Michael Unterkalmsteiner, Emelie Engström, and Markus Borg. An Industrial Case Study on Test Cases as Requirements. 2015.

[14] Elizabeth Bjarnason, Krzysztof Wnuk, and Björn Regnell. Requirements are slipping through the gaps - A case study on causes & effects of communication gaps in large-scale software development. In *RE 2011, 19th IEEE International Requirements Engineering Conference, Trento, Italy, August 29 2011 - September 2, 2011*, pages 37–46, 2011.

[15] J. Bosch. Achieving simplicity with the three-layer product model. *Computer*, 46(11):34–39, Nov 2013.

[16] Kevin Boudreau. Open platform strategies and innovation: Granting access vs. devolving control. *Management Science*, 56(10):1849–1872, 2010.

[17] P. Bourque, R. Dupuis, A. Abran, J.W. Moore, and L. Tripp. The guide to the Software Engineering Body of Knowledge. *IEEE Software*, 16(6):35–44, November 1999.

[18] S. Brad, M. Fulea, B. Mocan, A. Duca, and E. Brad. Software platform for supporting open innovation. volume vol.3, pages 224 – 9, Piscataway, NJ, USA, 2008.

[19] Roger J Calantone, S Tamer Cavusgil, and Yushan Zhao. Learning orientation, firm innovation capability, and firm performance. *Industrial marketing management*, 31(6):515–524, 2002.

[20] Pär Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering*, 7(3):139–151, 2002.

[21] H Chesbrough. Why companies should have open business models. *MIT Sloan management review*, 48(2), 2012.

[22] Henry Chesbrough, Wim Vanhaverbeke, and Joel West. *Open innovation: Researching a new paradigm*. Oxford university press, 2006.

[23] Henry Chesbrough, Wim Vanhaverbeke, and Joel West, editors. *New Frontiers in Open Innovation*. Oxford University Press, November 2014.

[24] Henry William Chesbrough. *Open innovation: the new imperative for creating and profiting from technology*. Harvard Business School Press, Boston, Mass., 2003.

[25] Sunita Chulani, Clay Williams, and Avi Yaeli. Software development governance and its concerns. In *Proceedings of the 1st International Workshop on Software Development Governance*, SDG 08, pages 3–6, New York, NY, USA, 2008. ACM.

[26] Massimo G. Colombo, Evila Piva, and Cristina Rossi-Lamastra. Open innovation and within-industry diversification in small and medium enterprises: The case of open source software firms. *Research Policy*, 2013.

[27] K. Conboy and L. Morgan. Beyond the customer: Opening the agile systems development process. *Information and Software Technology*, 53(5):535–42, May 2011.

[28] Kieran Conboy and Lorraine Morgan. Beyond the customer: Opening the agile systems development process. *Information and Software Technology*, 53(5):535 – 542, 2011.

[29] Kieran Conboy and Lorraine Morgan. Beyond the customer: Opening the agile systems development process. *Information and Software Technology*, 53(5):535–542, May 2011.

[30] Daniela S. Cruzes and Tore Dybå. Research synthesis in software engineering: A tertiary study. *Information and Software Technology*, 53(5):440 – 455, 2011.

[31] Daniela S Cruzes, Tore Dybå, Per Runeson, and Martin Höst. Case studies synthesis: A thematic, cross-case, and narrative synthesis worked example. *Empirical Software Engineering*, 2014.

[32] Linus Dahlander and Mats Magnusson. Relationships between open source software companies and communities: Observations from nordic firms. *Research Policy*, 34(4):481 – 493, 2005.

[33] Linus Dahlander and Mats Magnusson. How do firms make use of open source communities? *Long Range Planning*, 41(6):629 – 649, 2008.

[34] Linus Dahlander and Martin W. Wallin. A man on the inside: Unlocking communities as complementary assets. *Research Policy*, 35(8):1243 – 1259, 2006.

[35] Jorge Calmon de Almeida Biolchini, Paula Gomes Mian, Ana Candida Cruz Natali, Tayana Uchôa Conte, and Guilherme Horta Travassos. Scientific research ontology to support systematic review in software engineering.

Advanced Engineering Informatics, 21(2):133 – 151, 2007. Ontology of Systems and Software Engineering; Techniques to Support Collaborative Engineering Environments.

[36] Paul M. Di Gangi and Molly Wasko. Steal my idea organizational adoption of user innovations from a user innovation community: A case study of dell ideastorm. *Decision support systems*, 48:303–312, 2009.

[37] Koen Dittrich and Geert Duysters. Networking as a means to strategy change: The case of open innovation in mobile telephony. *Journal of Product Innovation Management*, 24(6):510 – 521, 2007.

[38] Peter F. Drucker. *Post-Capitalist Society*. HarperBusiness, New York, reprint edition edition, April 1994.

[39] W. Ebner, J.M. Leimeister, and H. Krcmar. Community engineering for innovations: the ideas competition as a method to nurture a virtual community for innovations. *R&D Management*, 39(4):342 – 56, Sept. 2009.

[40] Henry Edison, Nauman Bin Ali, and Richard Torkar. Towards innovation measurement in the software industry. *Journal of Systems and Software*, 86(5):1390 – 1407, 2013.

[41] Göran Ekvall. Organizational climate for creativity and innovation. *European Journal of Work and Organizational Psychology*, 5(1):105–123, 1996.

[42] Ellen Enkel, Oliver Gassmann, and Henry Chesbrough. Open r&d and open innovation: exploring the phenomenon. *R&D Management*, 39(4):311–316, 2009.

[43] Neil A Ernst and Gail C Murphy. Case studies in just-in-time requirements analysis. In *Empirical Requirements Engineering (EmpiRE), 2012 IEEE Second International Workshop on*, pages 25–32. IEEE, 2012.

[44] Robert Feldt. Do system test cases grow old? In *Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on*, pages 343–352. IEEE, 2014.

[45] Robert G. Fichman. Going beyond the dominant paradigm for information technology innovation research: Emerging concepts and methods. *Journal of the Association for Information Systems*, 5(8):11, 2004.

[46] Arlene Fink. *The Survey Handbook*. Sage, 2nd edition, 2003.

[47] Samuel Fricker. Requirements value chains: Stakeholder management and requirements engineering in software ecosystems. In *Requirements Engineering: Foundation for Software Quality*, pages 60–66. Springer, 2010.

[48] Rosanna Garcia and Roger Calantone. A critical look at technological innovation typology and innovativeness terminology: a literature review. *Journal of product innovation management*, 19(2):110–132, 2002.

[49] Oliver Gassmann. Opening up the innovation process: towards an agenda. *R&D Management*, 36(3):223–228, 2006.

[50] Oliver Gassmann and Ellen Enkel. Towards a theory of open innovation: three core process archetypes. pages 1–18, 2004.

[51] Oliver Gassmann, Ellen Enkel, and Henry Chesbrough. The future of open innovation. *R&D Management*, 40(3):213–221, 2010.

[52] Crt Gerlec, Andrej Krajnc, Marjan Hericko, and Jan Boznik. Mining source code changes from software repositories. In *Software Engineering Conference in Russia (CEE-SECR), 2011 7th Central and Eastern European*, pages 1–5. IEEE, 2011.

[53] Daniel M German. The gnome project: a case study of open source, global software development. *Software Process: Improvement and Practice*, 8(4):201–215, 2003.

[54] T. Greenhalgh and R. Peacock. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *BMJ*, 331(7524):1064–1065, 2005.

[55] Endre Grøtnes. Standardization as open innovation: two cases from the mobile industry. *Information Technology & People*, 22(4):367–381, 2009.

[56] Kazuki Hamasaki, Raula Gaikovina Kula, Norihiro Yoshida, A. E. Cruz, Kenji Fujiwara, and Hajimu Iida. Who does what during a code review? datasets of oss peer review repositories. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 49–52. IEEE Press, 2013.

[57] Elad Harison and Heli Koski. Applying open innovation in business strategies: Evidence from finnish software firms. *Research Policy*, 39(3):351 – 359, 2010.

[58] Elad Harison and Heli Koski. Applying open innovation in business strategies: Evidence from finnish software firms. *Research Policy*, 39(3):351–359, 2010.

[59] Donald E. Harter, Mayuram S. Krishnan, and Sandra A. Slaughter. Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science*, 46(4):451–466, 2000.

[60] L.P. Hattori and M. Lanza. On the nature of commits. In *23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops, 2008. ASE Workshops 2008*, pages 63–71, September 2008.

[61] Petra Heck and Andy Zaidman. An analysis of requirements evolution in open source projects: recommendations for issue trackers. page 43. ACM Press, 2013.

[62] Joachim Henkel. Selective revealing in open innovation processes: The case of embedded linux. *Research Policy*, 35(7):953–969, 2006.

[63] Joachim Henkel. Champions of revealing-the role of open source developers in commercial firms. *Industrial and Corporate Change*, 18(3):435–471, December 2008.

[64] Joachim Henkel, Simone Schöberl, and Oliver Alexy. The emergence of openness: How and why firms adopt selective revealing in open innovation. *Research Policy*, September 2013.

[65] Jim Highsmith and Alistair Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001.

[66] Martin Höst, Klaas-Jan Stol, and Alma Orucevic-Alagic. Inner Source Project Management. *Springer*, 2014.

[67] James Howison, Keisuke Inoue, and Kevin Crowston. Social dynamics of free and open source team communications. In Ernesto Damiani, Brian Fitzgerald, Walt Scacchi, Marco Scotto, and Giancarlo Succi, editors, *Open Source Systems*, number 203 in IFIP International Federation for Information Processing, pages 319–330. Springer US, January 2006.

[68] Eelko K.R.E. Huizingh. Open innovation: State of the art and future perspectives. *Technovation*, 31(1):2 – 9, 2011.

[69] Watts S. Humphrey. *Managing the software process*. SEI Series in Software Engineering. Software Engineering Institute, 1989.

[70] S. Husig and S. Kohn. Open cai 2.0 - computer aided innovation in the era of open innovation and web 2.0. *Computers in Industry*, 62(4):407 – 13, 2011.

[71] M. Ivarsson and T. Gorschek. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, 16(3):365–95, 2011.

[72] Samireh Jalali and Claes Wohlin. Systematic literature studies: database searches vs. backward snowballing. In *2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement,(ESEM 2012)*, pages 29–38, 2012.

[73] Slinger Jansen, Sjaak Brinkkemper, Jurriaan Souer, and Lutzen Luinenburg. Shades of gray: Opening up a software producing organization with the open software enterprise model. *Journal of Systems and Software*, 85(7):1495 – 1510, 2012.

[74] Anders Jonsson and Gunilla Svingby. The use of scoring rubrics: Reliability, validity and educational consequences. *Educational Research Review*, 2(2):130–144, 2007.

[75] Lena Karlsson, Åsa G. Dahlstedt, Björn Regnell, Johan Natt och Dag, and Anne Persson. Requirements engineering challenges in market-driven software development: An interview study with practitioners. *Information and Software Technology*, 49(6):588 – 604, 2007.

[76] Mahvish Khurum, Tony Gorschek, and Magnus Wilson. The software value map: An exhaustive collection of value aspects for the development of software intensive products. *Journal of Software: Evolution and Process*, 25(7):711–741, 2013.

[77] Barbara Kitchenham, Pearl Brereton, and David Budgen. Mapping study completeness and reliability - a case study. In *Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, pages 126–135, 2012.

[78] Barbara A. Kitchenham, David Budgen, and O. Pearl Brereton. Using mapping studies as the basis for further research - a participant-observer case study. *Inf. Softw. Technol.*, 53(6):638–651, June 2011.

[79] Barbara A Kitchenham and Shari L Pfleeger. Personal opinion surveys. In *Guide to Advanced Empirical Software Engineering*, pages 63–92. Springer, 2008.

[80] Eric Knauss, Daniela Damian, Alessia Knauss, and Arber Borici. Openness and requirements: Opportunities and tradeoffs in software ecosystems. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 213–222. IEEE, 2014.

[81] Tufan Koc. Organizational determinants of innovation capacity in software companies. *Computers & industrial engineering*, 53(3):373–385, 2007.

[82] Tufan Koc and Cemil Ceylan. Factors impacting the innovative capacity in large-scale companies. *Technovation*, 27(3):105–114, 2007.

[83] Peter Kraljic. Purchasing must become supply management. *Harvard business review*, 61(5):109–117, 1983.

[84] H.L. Kundel and M. Polansky. Measurement of observer agreement. *Radiology*, 228(2):303, 2003.

[85] Mikko O.J. Laine. Using knowledge from end-users online for innovations: Effects of software firm types. volume 114 LNBIP, pages 70 – 78, Cambridge, MA, United states, 2012.

[86] Karim Lakhani and Jill A. Panetta. The principles of distributed innovation. SSRN Scholarly Paper ID 1021034, Social Science Research Network, Rochester, NY, October 2007.

[87] Karim R Lakhani and Eric von Hippel. How open source software works: free user-to-user assistance. *Research Policy*, 32(6):923 – 943, 2003.

[88] Soren Lauesen. *Software requirements: styles and techniques*. Pearson Education, 2002.

[89] Gwanhoo Lee and Weidong Xia. The ability of information systems development project teams to respond to business and technology changes: a study of flexibility measures. *European Journal of Information Systems*, 14:75–92, March 2005.

[90] Gwendolyn K. Lee and Robert E. Cole. From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development. *Organization Science*, 14(6):633–649, 2003.

[91] Sang-Yong Tom Lee, Hee-Woong Kim, and Sumeet Gupta. Measuring open source software success. *Omega*, 37(2):426 – 438, 2009.

[92] Josh Lerner and Jean Tirole. Some simple economics of open source. *The journal of industrial economics*, 50(2):197–234, 2002.

[93] Marvin B Lieberman and David Bruce Montgomery. *First-mover (dis) advantages: Retrospective and link with the resource-based view*. Graduate School of Business, Stanford University, 1998.

[94] Johan Linåker, Maria Krantz, and Martin Höst. On infrastructure for facilitation of inner source in small development teams. pages 149–163, 2014.

[95] Johan Linåker, Hussan Munir, Per Runeson, Björn Regnell, and Claes Schrewelius. A Survey on the Perception of Innovation in a Large Product-focused Software Organization. *6th International Conference on Software Business - ICSOB*, 2015.

[96] Frank van der Linden, Björn Lundell, and Pentti Marttiin. Commodification of industrial software: A case for open source. *IEEE Software*, 26(4):77–83, 2009.

[97] Juho Lindman, Matti Rossi, and Pentti Marttiin. Applying open source development practices inside a company. In *Open Source Development, Communities and Quality*, pages 381–387. Springer, 2008.

[98] Luis Lopez-Fernandez, Gregorio Robles, Jesus M. Gonzalez-Barahona, and Israel Herraiz. Applying social network analysis techniques to community-driven libre software projects. *International Journal of Information Technology and Web Engineering (IJITWE)*, 1(3):27–48, 2006.

[99] Katarina Lund and Mats Magnusson. The delicate coexistence of standardized work routines and innovation. In *Proceedings of the 19th International Product Development Management Conference*, Manchester, UK, June 2012.

[100] Charlotte Möller and Madeleine Wahlqvist. Critical Success Factors for Innovative Performance of Individuals-A. *Management*, 39(5):1155–1161.

[101] Lorraine Morgan, Joseph Feller, and Patrick Finnegan. Exploring inner source as a form of intra-organisational open innovation. Helsinki, Finland, 2011.

[102] Lorraine Morgan and Patrick Finnegan. Open innovation in secondary software firms: An exploration of managers perceptions of open source software. *Databased for advances in Information Systems*, 41(1):76–95, 2010.

[103] Barbara Moskal, Keith Miller, and LA King. Grading essays in computer ethics: rubrics considered helpful. *ACM SIGCSE Bulletin*, 34(1):101–105, 2002.

[104] David C. Mowery. Plus ca change: Industrial R&D in the third industrial revolution. *Industrial and Corporate Change*, 18(1):1–50, 2009.

[105] Neeshal Munga, Thomas Fogwill, and Quentin Williams. The adoption of open source software in business models: A red hat and ibm case study. pages 112 – 121, Vanderbijlpark, Emfuleni, South africa, 2009.

[106] Hussan Munir and Per Runeson. Software testing in open innovation: An exploratory case study of the acceptance test harness for jenkins. ICSSP 2015, pages 187–191, New York, NY, USA, 2015. ACM.

[107] Hussan Munir, Krzysztof Wnuk, and Per Runeson. Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering*, 2015.

[108] Abhishek Nirjar. Accruing innovation in software firms through employees commitment. *International Journal of Indian Culture and Business Management*, 6(4):391–409, January 2013.

[109] Lucas D Panjer, Daniela Damian, and Margaret-Anne Storey. Cooperation and coordination concerns in a distributed software development project. In *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, pages 77–80. ACM, 2008.

[110] Vinit Parida, Mats Westerberg, and Johan Frishammar. *Effect of open innovation practices on SMEs innovative performance: An empirical study*. 2011.

[111] Ralph Peters and Andy Zaidman. Evaluating the lifespan of code smells using software repository mining. pages 411–416. IEEE, March 2012.

[112] Kai Petersen and Nauman Bin Ali. Identifying strategies for study selection in systematic reviews and maps. In *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*, pages 351–354, 2011.

[113] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering*, volume 17, page 1, 2008.

[114] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[115] CK Prahalad and MS Krishnan. The new meaning of quality in the information age. *Harvard business review*, 77(5):109–18, 1998.

[116] Shaowen Qin. Managing process change in software organizations: Experience and reflection. *Software Process: Improvement and Practice*, 12(5):429–435, 2007.

[117] T. Rayna and L. Striukova. Large-scale open innovation: open source vs. patent pools. *International Journal of Technology Management*, 52(3-4):477 – 96, 2010.

[118] S.C. Reid. BS 7925-2: the software component testing standard. In *Proceedings First Asia-Pacific Conference on Quality Software*, pages 139–148, 2000.

[119] G. Robles, S. Koch, and J.M. Gonzalez-Barahona. Remote analysis and measurement of libre software systems by means of the CVSAnalY tool. "W15S Workshop - 26th International Conference on Software Engineering, pages 51–5. IEEE, 2004.

[120] R. Rohrbeck, K. Holzle, and H.G. Gemunden. Opening up for competitive advantage - how deutsche telekom creates an open innovation ecosystem. *R & D Management*, 39(4):420 – 30, 2009.

[121] Bertil Rolandsson, Magnus Bergquist, and Jan Ljungberg. Open source in the firm: Opening up professional practices of software development. *Research Policy*, 40(4):576 – 587, 2011.

[122] Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering - Guidelines and Examples*. Wiley, 2012.

[123] Per Runeson and Mats Skoglund. Reference-based search strategies in systematic reviews. In *Proceedings 13th International Conference on Empirical Assessment & Evaluation in Software Engineering (EASE)*, Durham University, UK, 2009. British Computer Society.

[124] Walt Scacchi. Understanding the requirements for developing open source software systems. In *Software, IEE Proceedings-*, volume 149, pages 24–39. IET, 2002.

[125] Walt Scacchi. *Understanding requirements for open source software*. Springer, 2009.

[126] Walt Scacchi. Collaboration practices and affordances in free/open source software development. In *Collaborative software engineering*, pages 307–327. Springer, 2010.

[127] Leif Singer, Norbert Seyff, and Samuel A. Fricker. Online social networks as a catalyst for software and IT innovation. In *4th International Workshop on Social Software Engineering, SSE'11, September 5, 2011 - September 5, 2011*, SSE'11 - Proceedings of the 4th International Workshop on Social Software Engineering, pages 1–5, 2011.

[128] Dag I. K. Sjøberg, Tore Dybå, and Magne Jørgensen. The future of empirical methods in software engineering research. In *Workshop on the Future of Software Engineering (FOSE 2007)*, pages 358–378, 2007.

[129] Dag I. K. Sjøberg, Jo Erskine Hannay, Ove Hansen, Vigdis By Kampenes, Amela Karahasanovic, Nils-Kristian Liborg, and Anette C. Rekdal. A survey of controlled experiments in software engineering. *IEEE Trans. Software Eng.*, 31(9):733–753, 2005.

[130] Daniel Ståhl and Jan Bosch. Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, 87:48–59, 2014.

[131] Wouter Stam. When does community participation enhance the performance of open source software companies? *Research Policy*, 38(8):1288 – 1299, 2009.

[132] Matthias Stuermer, Sebastian Spaeth, and Georg Von Krogh. Extending private-collective innovation: a case study. *R&D Management*, 39(2):170–191, 2009.

[133] Han van der Meer. Open innovation the dutch treat: Challenges in thinking in business models. *Creativity and Innovation Management*, 16(2):192–202, 2007.

[134] Kris Ven and Herwig Mannaert. Challenges and strategies in the use of open source software by independent software vendors. *Information and Software Technology*, 50(9):991–1002, 2008.

[135] G. von Krogh and S. Spaeth. The open source software phenomenon: characteristics that promote research. *Journal of Strategic Information Systems*, 16(3):236 – 53, 2007.

[136] Georg Von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, 32(7):1217 – 1241, 2003.

[137] Yonggui Wang, Ruqiong Tong, and Shanji Yao. An empirical study on external influencing factors of user innovation performance. In *Management and Service Science, 2009. MASS '09. International Conference on*, pages 1–4, 2009.

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

[139] J. West and S. Gallagher. Challenges of open innovation: the paradox of firm investment in open-source software. *R & D Management*, 36(3):319 – 31, 2006.

[140] Joel West. How open is open enough?: Melding proprietary and open source platform strategies. *Research Policy*, 32(7):1259 – 1285, 2003.

[141] Joel West and Marcel Bogers. Leveraging external sources of innovation: A review of research on open innovation. *Journal of Product Innovation Management*, 2013.

[142] Joel West and Marcel Bogers. Leveraging external sources of innovation: A review of research on open innovation. *Journal of Product Innovation Management*, 2013.

[143] Joel West and Scott Gallagher. Challenges of open innovation: the paradox of firm investment in open-source software. *R&d Management*, 36(3):319–331, 2006.

[144] Joel West and David Wood. Creating and evolving an open innovation ecosystem: Lessons from symbian ltd. *Available at SSRN 1532926*, 2008.

[145] Joel West and David Wood. Evolving an open ecosystem: The rise and fall of the symbian platform. *Advances in Strategic Management*, 30:27–67, 2013.

[146] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, 11(1):102–107, 2006.

[147] K. Wiklund, D. Sundmark, S. Eldh, and K. Lundvist. Impediments for automated testing – an empirical analysis of a user support discussion board. In *Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on*, pages 113–122, 2014.

[148] K. Wnuk, D. Pfahl, D. Callele, and E. Karlsson. How can open source software development help requirements management gain the potential of open innovation: an exploratory study. pages 271 – 9, Piscataway, NJ, USA, 2012.

[149] Krzysztof Wnuk, Dietmar Pfahl, Davide Callele, and Even André Karlsson. How can open source software development help requirements management gain the potential of open innovation: an exploratory study. In *Proceedings of the 2012 6th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 271 – 9, Piscataway, NJ, USA, 2012.

[150] Krzysztof Wnuk and Per Runeson. Engineering open innovation–towards a framework for fostering open innovation. pages 48–59. Springer, 2013.

[151] Krzysztof Wnuk, Per Runeson, Matilda Lantz, and Oskar Weijden. Bridges and barriers to software ecosystem participation - a case study. *Information and Software Technology*, 56(11):1493–1507, 2014.

[152] Claes Wohlin and Rafael Prikladnicki. Systematic literature reviews in software engineering. *Information & Software Technology*, 55(6):919–920, 2013.

[153] Marvin V Zelkowitz, Dolores R Wallace, and D Binkley. Culture conflicts in software engineering technology transfer. In *NASA Goddard Software Engineering Workshop*, page 52, 1998.