An Empirically Based Theory for Open Software Engineering Tools

Munir, Hussan

2018

Document Version:
Publisher's PDF, also known as Version of record

An Empirically Based Theory for Open Software Engineering Tools

Hussan Munir

Doctoral Dissertation, 2018
Department of Computer Science
Lund University

LU-CS-DISS 2018-2
Doctoral Dissertation 59, 2018

ISBN: 978-91-7753-738-0 (Printed)
ISBN: 978-91-7753-739-7 (Electronic)
ISSN: 1404-1219

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, 2018

ABSTRACT

Many companies and developers from OSS communities create open tools collaboratively in which software developers improve upon the code and share the changes within the community. Open tools (e.g., Jenkins, Gerrit, and Git) offer features or performance benefits that surpass their commercial counterparts in the core product development. Participation in OSS tools communities greatly dismantled the closed innovation model and lured organizations towards Open Innovation (OI). Harnessing the external knowledge that OI offers, requires better understanding regarding what to develop internally and what to acquire from outside the organization, how to cooperate with potential competitors, and when to conceal or reveal code while working with OSS communities.

The aim of this thesis is to investigate how software-intensive organizations utilize the external and internal knowledge from OSS tools communities using Open Innovation to improve their core product development. First, this aim was achieved by exploring and reporting the existing evidence of OI in software engineering. Second, by providing a solution for software-intensive organizations regarding how to choose the right level of openness while working with OSS tools communities. Finally, we validated the proposed solution in multiple organizations.

The thesis followed an empirical approach by conducting a systematic mapping study, case study, design science based contribution acceptance model, theory creation and validation of the theory. First, we conducted a systematic mapping study to synthesize the existing evidence on OI in software engineering and identified the research gaps. Second, we conducted an exploratory case study at Sony Mobile to explore how a software organization uses OSS tools communities to facilitate its core product development. Third, we proposed a theory of openness for organizations which provides guidelines regarding how to work with OSS tools communities. Fourth, we presented a contribution acceptance model and metamodel to assist strategic product planning in what to develop internally and what to share as OSS in the proprietary products.. Finally, we validated the proposed theory of openness for tools in two automotive companies by conducting focus groups.

The main conclusion of the thesis is that software-intensive organizations need to acquire external knowledge from OSS tools communities to accelerate their internal innovation process. Improved and flexible development tools provide opportunities to shorten the development time, improves new product releases and upgrades, frees up developers time, increased quality assurance, sharing the maintenance cost and steer communities to facilitate organization's business models. However, it can only be achieved if there are well-defined guidelines for developers and managers to operationalize working with OSS tools communities. This thesis presents a theory of openness to facilitate managers on how to works with OSS tools communities. The theory suggests that the top management needs to develop new roles and legal procedures to educate developers regarding how to use and contribute to OSS tools communities for a faster development environment. This openness provides opportunities for the organizations to reduced development cost, shorten development time and process and product innovation.

POPULAR SUMMARY

IT IS MORE BLESSED TO GIVE THAN TO RECEIVE

HUSSAN MUNIR, DEPARTMENT OF COMPUTER SCIENCE

Open Innovation (OI) allows knowledge flow both inside-out and outside-in the company, and may or may not be attached to monetary transactions.

OI penetrates several industries, as many companies discover that their business may benefit from sharing knowledge with other companies. The use of proprietary tools for software development has several drawbacks, e.g. expensive licensing costs, lack of customizability, delayed implementation of requirements, the inability of fixing things in-house, and difficulty in finding solutions that meet current needs. On the other hand, the use of open tools for software development in the companies is an area which companies apply OI principles to. By using open tools for software development companies share the innovation cost and rewards and also risks.

Why should companies open up?

Software companies cannot afford to work in a closed way due to the continuous need for automation and increased speed. Developing tools internally for software development may entail significant costs and companies may miss the latest trends in OSS tools ecosystem. Therefore, companies need to tag along with open tools communities to extract the external knowledge in a timely manner. From two research studies, we distilled a set of triggers that drive companies towards applying OI strategies in sharing their tools openly. The triggers include aspects of access to workforce, development speed, reduced license costs, work-flow flexibility, maintenance costs and increased quality assurance.

The key findings of this research entail how software companies may choose the right level of openness in their proprietary products and open tools used for the development of company's internal products. First, the contribution acceptance

model is presented for companies, which assists in what to develop internally and what to share and take from open source software. Second, the theory of openness helps organizations how to develop and use open tools communities. We have presented different strategies for companies to choose the right level of openness. While working with open tools, it is paramount to share the source code in order to avoid getting trapped into internal maintenance and integration cost. Companies should strive for standard solutions and reduce the number of variants of open tools by contributing their source code towards open source communities.

Implications for companies

Software companies use OSS tools communities as an implementation of OI to create business value for their core products. Therefore, they may achieve OI by choosing the right level of openness. Companies may assign dedicated resources to work with open tools communities, with the goal to acquire and assimilate knowledge in the company's core product development.

In order to create new open tools communities and steer them towards the company's business model, companies need to invest more of their employees' time in open source communities. Then they may gain advantages, such as access to skilled resources, better continuous integration integration process, faster upgrades and releases, reduced development time, and share the maintenance cost with other developers in the open source communities.

to my father, the most honest man I know,
to my mother, the most patient and selfless lady I know,
to my sister, for all the support, guidance and love,
to my brother, for his insights and humor in growing up together

"*There is no beauty better than intellect*" **- Prophet Muhammad (PBUH)**

ACKNOWLEDGEMENTS

This work was funded by Synergies project, grant 621-2012-5354 from the Swedish Research Council and partly funded by the EASE industrial excellence center.

First praise is to Allah, the Almighty, on whom ultimately I depend for sustenance and guidance. You have given me the power to believe in my passion and pursue my dreams. I could never have done this without the faith I have in you, the Almighty.

My sincerest thanks are extended to my supervisors Prof. Dr. Per Runeson and Dr. Krzysztof Wnuk for the continuous support of my Ph.D. study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my Ph.D. study.

The research presented in this thesis was conducted in close cooperation between academia and industry. Therefore, I am particularly grateful to Sony Mobile and two anonymous case companies. I am also thankful to all of the Department of Computer Science faculty members and the Software Engineering Research Group for their support and research collaborations.

Finally, this journey would not have been possible without the support of my family. I am eternally grateful for encouraging me in all of my pursuits and inspiring me to follow my dreams. I always knew that you believed in me and wanted the best for me. Thank you for teaching me that my job in life was to learn, to be happy, and to know and understand myself; only then could I know and understand others. You are indispensable. Heartfelt thank 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, (2016) 21: 684-723.

II **Open Innovation using Open Source Tools: A Case Study at Sony Mobile**
Hussan Munir, Johan Linåker, Krzystof Wnuk, Per Runeson and Björn Regnell, *Empirical Software Engineering*, (2018) 23: 186-223.

III **A Theory of Openness for Software Engineering Tools in Software Organizations**
Munir, Hussan, Per Runeson, and Krzysztof Wnuk.
Information and Software Technology,(2018) 97: 26-45.

IV **Motivating the Contributions: An Open Innovation perspective on What to Share as Open Source Software**
Linåker, Johan, Hussan Munir, Krzysztof Wnuk, and Carl Eric Mols.
Journal of Systems and Software, (2018) 135: 17-36.

V **Open Tools for Software Engineering using the Theory of Openness : A Validation Study in the Automotive Industry**
Hussan Munir, Per Runeson, Krzystof Wnuk and Johan Linåker, *Submitted to ESEM 2018.*

Related Publications

VI **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, pp 66-80.

VII **Software Testing in Open Innovation: An Exploratory Case study of Acceptance Test Harness for Jenkins**
Hussan Munir, Per Runeson
International Conference on Software and System Process (ICSSP), 2015 , pp 187-191.

Contribution statement

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

Paper I
Hussan Munir is the lead author responsible for the designing the research plan and executing the study followed by a validation and paper review from Dr. Krzysztof Wnuk and Prof. Per Runeson. Hussan Munir was responsible for data collection, analysis and writing the paper.

Paper II
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. Krzysztof Wnuk was also involved in conducting the interviews with industry professionals.

Paper III
Hussan Munir is the lead author responsible for the research design and literature analysis process in the creation of theory. Prof. Per Runeson and Dr. Krzysztof Wnuk were involved in giving the feedback in all phases of the paper.

Paper IV
Johan Linåker is the lead author together with Hussan Munir and Dr. Krzysztof Wnuk. I was responsible for designing, executing, validating and writing the research work with other authors.

Paper V
Hussan Munir proposed the idea of using repertory grid analysis and designed the validation study to conduct the workshops at the case companies. Prof. Per Runeson and Dr. Krzysztof Wnuk were involved in giving feedback in designing and writing this paper. However, the workshops were conducted by Hussan Munir and Prof. Per Runeson at the case companies.

CONTENTS

INTRODUCTION

1 Introduction

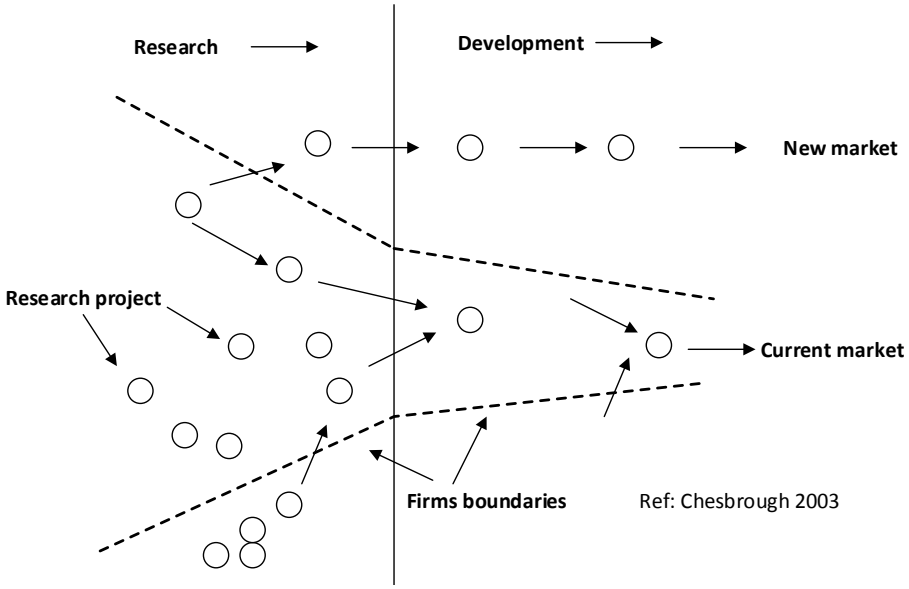
The rising cost and the increased demand for delivering products with the faster time to market have put extensive pressure on many organizations [27]. Software-intensive organizations (SIO) are constantly reconsidering which strategies are successful in generating ideas and bringing them to market. These strategies entail harnessing external ideas while leveraging their in-house R&D outside their current operations [31]. Organizations struggle to remain competitive using the existing models of innovation and need a shift in the ways of working by combining the internal ideas with external ideas. One possible way to reduce the development cost and shorter time to market is to use Open Innovation (OI) to harvest the external ideas.

OI is an emerging management paradigm which originated from high technology industry practices in the US and Japan [30]. OI can be traced back to Allen's [6] collective invention in 1980's. Two decades after Allen's paper from 1983, Henry Chesbrough [30] coined the term *Open Innovation* as *"a distributed innovation process based on purposively managed knowledge flows across organizational boundaries, using pecuniary and non-pecuniary mechanisms in line with the organization's business model"*. This phenomenon is explained with the help of Fig.1. The dotted line in the funnel shows the boundary of the company where ideas can seep in and out. The bubbles represent the research projects and arrows highlight the flow of ideas in and out of the companies. Furthermore, the vertical line in Fig.1 separates the research phase from the development phase of the company. Ideas can originate from inside the company's research process, but some of the ideas may seep out of the companies, either in the research phase or later in the development phase. These innovative ideas are utilized by companies to create a new market or make use of the existing market. One typical example 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 move inside. Google's Android was taken in by companies like Sony and Samsung to adapt it in a way

which was more in line with their business model and thus a clear case of utilizing an external project to access the existing Android market.

OI initiated an unabated interest among researchers in innovation management [83], economics, psychology, sociology, and also Software Engineering [182]. The work initiated by Chesbrough inspired both scholars and practitioners to rethink the design of the innovation strategies in a networked environment [83]. 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) [66].

Figure 1: Open Innovation



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 [133]. In response, Chesbrough undercuts the logic of the *Closed Innovation* model of R&D and developed the logic of the *Open Innovation* model due to the changed conditions under which organizations innovate. For example, the rise of the internet has made the knowledge access and sharing capabilities easier using Open Source Software (OSS) [30].

In the field of software engineering, the success of *Open Source Software (OSS)* indicates its existence before the term OI was coined [112]. The introduction of OSS in commercial settings have opened up new possibilities for innovation in software-intensive organizations. This shift towards openness indicates that the

internal R&D is no longer the only strategic asset for the companies in creating products and services. Access to, and interplay with, external sources and actors provide not only new opportunities but also create new challenges. One specific type of OSS is software engineering tools used in the development of software-intensive products. The tools themselves are not the source of revenue for the software-intensive organizations, but they rely heavily on them to improve the software development process. Further, the costs of improving the tools and keeping them up to date may be significant, and thus software-intensive organizations may want to share the costs with other organizations [27].

However, it should be noted that OSS is *not* equivalent to OI. OSS is used as an example of OI in the studies included in this thesis [29]. 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, there is a distinction between OI and OSS. First, OSS and OI may differ by using different intellectual property rights (IPR) strategies. For example, when IBM created the Eclipse platform, they invited competing companies to cooperate in an OI ecosystem [187]. In OI, companies may retain the ownership of IPR as oppose to OSS. Secondly, companies leading OI complement their internal closed innovation process by acquiring external knowledge [137]. Thirdly, OI companies have a business model influenced by the definition of OI, where differential assets are kept secret to create value. Therefore, the degree of openness lies in the hands of the companies in relation to OSS communities. Finally, companies try to govern and steer open tools platform to facilitate their internal product development by co-developing development tools with other stakeholders in the ecosystem. Therefore, OSS is a natural way of *implementing* OI in software-intensive product development organizations, where OSS communities act as innovation catalyst.

Another example of OI 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 a significant amount of money in the Linux development, other firms such as Nokia, Hitachi, and Intel also made substantial investments as well [110]. 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 [30].

As OSS matured and became commercially viable to deliver high-quality products, software-intensive organizations started using them for the development of their proprietary products in two possible scenarios. First, when an organization decides to release proprietary code as OSS and create a community or ecosystem to improve its internal product. Second, when an organization wants to use OSS code for tools or for the code of the product. In this thesis, we proposed Contribution Acceptance Process (CAP) model and Theory of openness for tools to address both scenarios.

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. [Paper I] 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 [83, 186, 196] or focused on the metrics used to measure innovation in OSS [50].

Organization use different strategies to engage in OSS tools communities [41], e.g. adopting selective revealing [76] or OI models [27]. West et al. [188] 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. [176] 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. Bosch [23] claims that speed, data and ecosystems are the main factors that impact software-intensive organizations in their software engineering practices. At the same time, the size of software-intensive products continues to grow. This growth incurs the need for faster and better adoption of applications, technologies, components, services, and ecosystem partners. In order to address this challenge, software-intensive organizations may utilize OSS tools communities to increase the speed, reduce the development and maintenance costs.

In addition, OI entails challenges on process and business levels. West et al. [191] 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. Software-intensive organizations intending 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 entail different modes of working in terms of dedicated resouces [108, 194] and OSS governance mechanisms [110] to facilitate software development in an OI context. Dahlander [42] concluded that initiating an OSS project is often a pragmatic way of attracting the 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 the organization's business model. Van der Linden et al. [120] concluded that when a software product loses its competitive value in terms of profitability, customers, innovation and learning [97] with the passage of time due to improvements and ever-growing size of the software, it becomes a good candidate for OSS development.

Table 1: Definitions

Terms	**Definition**
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].
Software-intensive product organization	It refers to organizations developing products or services with a substantial amount of software defining the product/service behavior, mostly embedded in physical products.

However, the shift from the *Closed innovation* to the *Open innovation* model poses significant challenges to software-intensive organizations in terms of when to conceal and when to reveal in relation to their competitors. The openness challenges software-intensive organizations on both operational and strategic levels. This thesis focuses on investigating the OSS tools communities considered repre-

sentative examples of OI to investigate the impacts of OI on firms core product development. Particularly, the triggers for software-intensive organizations to utilize the OSS tools communities and the innovation outcomes attached to it. Furthermore, the thesis proposes a theory of openness which provides guidelines for software-intensive organizations to make strategic decisions regarding *OSS tools* (e.g., Jenkins and Gerrit), which are not the core business (non-pecuniary) for the organization but are vital to support the internal product development. The definition of the terms used in the thesis can be seen in Table 1.

3 Research goals

The overall aim of the thesis is to better understand OI in software engineering, thus the following *Research Goals (RG)* are formulated.

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

RG2: To explore how software-intensive development organizations use OSS tools as an enabler for OI and innovation outcomes.

RG3: To provide strategic guidelines for managers regarding when and how to be open in relation to OSS tools and proprietary products.

RG4: To validate the strategic guidelines in relation to *RG3* with practitioners working in the software-intensive development organizations.

Figure 2 provides an overview of the research process. 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. The outcome of *Paper I* is the literature review.

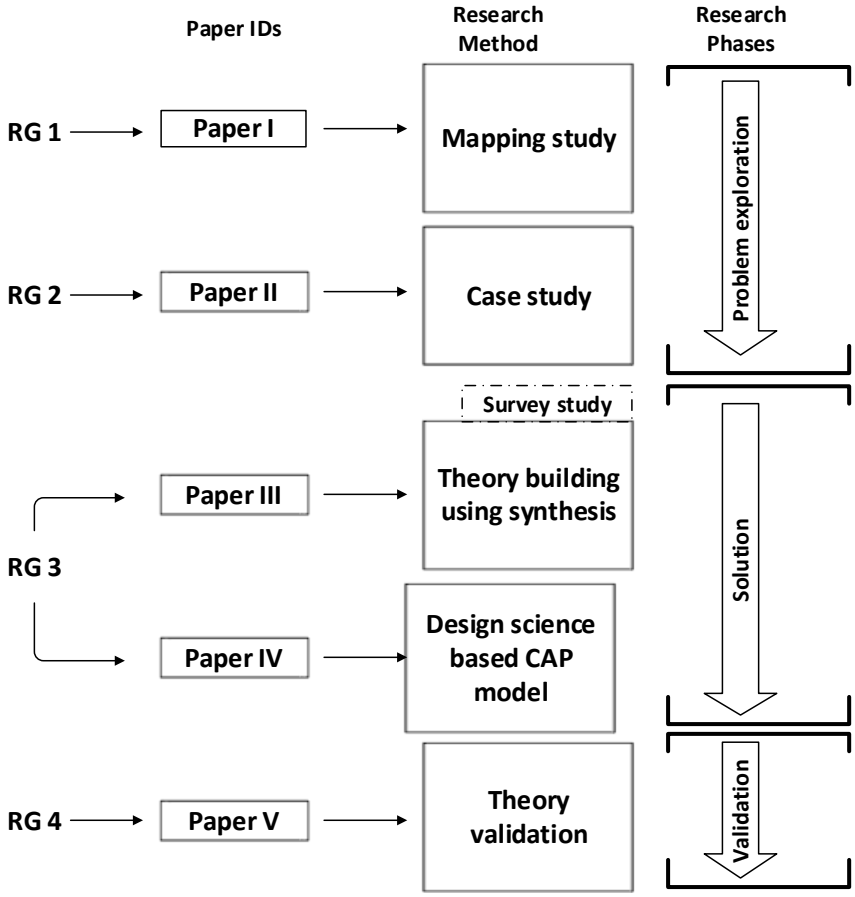
RG2 is relevant to investigate OI on the use of OSS tools in a sofwtare product development and influenced by *RG1*. The literature lacks evidence about the performance of OI on the fined grained product development level [30]. *Paper II* is aimed at exploring why and how a software-intensive organization adopts OI using OSS tools communities. In addition, *Paper II* also points out the innovation outcomes gained by the case organization. The outcome of *Paper II* is the detailed case study understanding of OI in a software-intensive company.

RG3 leads to two solution papers. *Paper III* aims to define support for strategic decisions in software organizations in relation to OSS tools and their impact on core product development. *Paper IV* investigates strategic decisions for software-intensive organizations on the core product level. However, the focus of this thesis remains on the use of OSS tools in organization's internal product development.

The outcome of *Paper III* and *Paper IV* is the theory of openness and contribution acceptance process model respectively.

RG4 leads to *Paper V*, which is a validation study for the theory of openness in two automotive companies.

Figure 2: Research overview and mapping of RGs to papers



4 Research methodology

Several research methods were utilized to meet the research goals. The thesis mainly consists of exploratory and evaluative empirical research [193], based on a

systematic mapping study [147], survey [59] and case study [159] research method (see Table 2).

Table 2: Research strategy used for each paper based on pp. 15 [159]

Paper Id	Research Objective	Research Strategy	Design Type
Paper I	Exploratory	Systematic mapping study	Flexible
Paper II	Exploratory	Case study	Flexible
Paper III	Solution	Theory building and Survey	Flexible + Fixed
Paper IV	Solution	Case study using design science principles	Flexible
Paper V	Validation	Case study	Flexible

Paper I presents a systematic mapping study designed to explore the OI literature on software engineering. Prior reviews were either not systematic [83, 196], partly systematic [186] or, for example, focus on the history or evolution of OSS or available innovation metrics [50]. 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 [98]. Therefore, a systematic mapping study was chosen over the systematic literature review in order to explore the OI notion in software engineering.

Paper II presents a case study, which not only investigates OI in an exploratory manner but also makes an attempt to evaluate OI performance in a software-intensive organization. The research questions in *Paper II* are partly based on the findings from *Paper I*. First, the study explores the top contributors to 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 reposi-

tories 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.

The case company used in the studies is Sony Mobile and the units of analysis 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 case study is responsible for the development of Android phones. Furthermore, Sony is becoming more and more open in terms of using OSS communities. Jenkins and Gerrit are OSS examples studied in the paper seen as an enabler for OI in software engineering.

Paper III aims at synthesizing a theory of openness for software engineering tools in software organizations, aimed to guide managers in defining more efficient strategies towards open tools communities. We synthesize empirical evidence from a systematic mapping study [Paper I], a case study [Paper II], and a survey, using a narrative synthesis method [Paper III]. The survey questionnaire was distributed among 500 employees working for software-intensive organizations using Gerrit, Jenkins or Git communities in their development or also, contributing to those communities. We extracted the email list of Jenkins, Gerrit and Git communities from GitHub and distributed the survey among all contributors and non contributors having organizational affiliations in their email addresses. The synthesis method entails four steps: (1) Developing a preliminary synthesis, (2) Exploring the relationship between studies, (3) Assessing the validity of the synthesis, and (4) Theory formation. The final step in the synthesis method proposed theory of openness for software engineering tools, according to the theory-building framework proposed by Sjøberg et al. [169]. The theory consists of 1) constructs, 2) propositions, and 3) explanation in Paper III.

Paper IV is a case study designed based on the design science approach [81]. The work was initiated by problem identification and analysis of its relevance at Sony Mobile. This was followed by an artifact design process where the artifacts (the CAP model and information meta-model) were proposed and validated at Sony Mobile. We conducted informal consultations with four experts at Sony Mobile who is involved in the decision-making process of OSS contributions. Simultaneously, internal processes and policy documentation at Sony Mobile were studied. Next, we accessed the additional data sources acquire the contribution repositories. All these steps were performed in close academia-industry collaboration between the researchers and Sony Mobile.

Paper V aims to validate the theory of openness by performing a repertory grid analysis [95] using focus groups [159] in two companies from the automotive industry. Kelly proposed the personal construct theory (PCT) and the associated repertory grid technique to elicit and analyze these personal constructs [95][Paper III]. The grid is comprised of following three basic concepts: 1) Elements Elicitation, 2) Constructs Elicitation and 3) Ratings. There are two essential ways to select grid elements: a) elicit elements from participants, b) provide participants

with elements. This study chooses to the provide elements participants since the objective was to learn more about the specific set of elements derived from the theory of openness [Paper III].

First, the participants in the focus group were given an introduction to constructs and elements to develop a common understanding in the whole group. Second, participants from company A picked *Jenkins* and participants from company B selected an internal tool entitled Awesome framework for the focus group. Third, a survey link was distributed among all the participants to rate each element against the constructs based on the selected tools from their internal development environment. Fourth, we held a discussion among participants based on the ratings to further explore their ratings. The discussion part was recorded and transcribed to further explore the rationales for the participant's ratings. Finally, repertory grid analysis and focus groups were used to validate the propositions derived from the theory of openness [Paper III].

5 Results and synthesis

This section summarizes the results from the papers included in this thesis. For each paper, we state the rationale, the methodology used (see table 2) and the key findings.

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

Paper I identifies 33 studies, divided into nine themes as a result of thematic analysis [38]. 17 out of 33 studies were conducted with high rigor and in an industrial relevant context. The key themes identified in the study are as follow:

1. Intellectual properties 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 [159] and paper type classification [193] followed by the rigor and relevance analysis [85]. Twenty 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 external knowledge into their R&D activity have a better chance of gaining financial advantage. Furthermore, an important implication for an 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 explore and evaluate how a software-intensive organization uses OSS as an enabler for OI and gains benefits

Paper II 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 extract and assimilate external knowledge using OSS tools 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 the *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. 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. This transition from closeness to openness is driven bottom-up from the engineers at Sony Mobile. 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.

The Tools department was 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 II further

explores if there are any innovation outcomes attached to these tools and identified the following innovation outcomes 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.

RG3: To provide a theory for managers regarding when and how to be open in relation to development outcomes and development process

Paper III presents a *theory of openness for software engineering tools* in software organizations that complement and expands our previous research efforts [Paper I][Paper II] and provides the necessary organizational aspects that support software-intensive organizations in their transformation towards OI. The increased use of Open Source Software (OSS) affects how software-intensive product development organizations innovate and compete, moving them towards Open Innovation (OI). Specifically, software engineering tools have the potential for OI, but require better understanding regarding what to develop internally and what to acquire from outside the organization, and how to cooperate with potential competitors.

However, we have found no guidelines for software-intensive organizations in order to make strategic decisions regarding *OSS tools*, i.e. what role in Huizing's

taxonomy to choose in the open innovation (i.e. open processes, open outcomes), for OSS tools which are not the core business (non-pecuniary) for the organization (e.g., OSS tools like Jenkins and Gerrit) but are vital to support the internal product development. The scope of this study covers the use of non-pecuniary OSS tools in organizations' proprietary software development for outside-in and inside-out innovation (i.e. coupled innovation). Furthermore, the study focuses on the strategic role of OSS tools in an organization, where we use software build tools as cases, due to their strategic role in the build chain [Paper II][Appendix D].

The theory of openness for OSS tools in software engineering presents four constructs: (1) Strategy, (2) Triggers, (3) Outcomes, and (4) Level of openness. We synthesize the theory from two previous empirical studies [Paper I][Paper II] complemented by a survey in the Git, Gerrit and Jenkins communities [Appendix D]. The theory presents four classes of openness in companies with their respective focus:

1. Laggards – Routine business
2. Leverage – Resource optimization
3. Lucrativeness – Acting as a think-tank
4. Leaders – Growth through ecosystems

Each category has the different levels of openness, based on their strategies (proactive or reactive) in relation to goals (cost saving or inspirational). First, *laggards* respond to paradigm shifts and all strategies are reactive, aiming to reduce the development cost (i.e. integration). Second, in *leverage* category, organizations use the external sources of innovation by inspiring their internal developers to participate in various OSS tools communities, prior to internal R&D work. It not only adds to product and process innovation but also inspires developers to exchange ideas on discussion forums to develop competence. Third, *Lucrativeness* deals with investing in existing OSS communities to be able to influence and steer these communities in the same direction as the organizational interests. The objective is to support internal innovation and reduce costs by investing in OSS tools communities. The use of OSS tools communities helps organizations to reduce time-to-market. Fourth, *Leaders* are organizations that focus on creating new communities and ecosystems to strengthen their business model.

The theory provides *strategic* guidelines and helps software-intensive organizations to adopt OI tools in relation to reduced development cost, shorter time-to-market and process, and product innovation. The theory reasons that openness provides opportunities to reduce the development cost and development time. Furthermore, OI positively impacts on the process and product innovation, but it requires investment by organizations in OSS communities. By betting on openness, organizations may be able to significantly increase their competitiveness but it requires management's support.

Paper IV proposes a Contribution Acceptance Process (CAP) model and meta model. The model helps software-intensive product development organizations to classify artifacts, such as features, plug-ins, or complete projects, according to business impact (low to high) and control complexity (low to high). Business impact refers to the profit from the artifact, and control complexity refers to the difficulty in acquiring and controlling the technology. An artifact is categorized into the following four categories where each category represents a specific artifact type with certain characteristics and contribution strategy.

- Strategic artifacts: high business impact and high control complexity.
- Platform/leverage artifacts: high business impact and low control complexity.
- Products/bottlenecks artifacts: low business impact and high control complexity.
- Standard artifacts: low business impact and low control complexity.

In turns, organizations may estimate and plan whether an artifact should be contributed or not. Open Source Software (OSS) ecosystems have reshaped the ways how software-intensive organizations develop products and deliver value to customers. However, organizations still need support for strategic product planning in terms of what to develop internally and what to share as OSS. Existing models accurately capture commoditization in the software business, but lack operational support to decide what contribution strategy to employ in terms of what and when to contribute. Further, an information meta-model is proposed that helps operationalize the CAP model at the organization. In a design science influenced case study executed at Sony Mobile, the CAP model was iteratively developed in close collaboration with the experts from Sony Mobile. The CAP model provides an operational OI perspective on what firms involved in OSS ecosystems should share, by helping them motivate contributions through the creation of contribution strategies. The goal is to help maximize return on investment and sustain needed influence in OSS ecosystems.

Static validation was done through continuous consultations with experts at Sony Mobile for the CAP-model and its related information meta-model. In these consultations, the models were discussed and improvement ideas were collected and used for iterative refinement and improvement. Experts from Sony Mobile were asked to run the CAP model against examples of features in relation to the four software artifact categories and related contribution strategies that CAP model describes. The examples of how the CAP model and meta-model are used is further presented in *Paper IV*. These examples help to evaluate functionality, completeness, and consistency of the CAP model and associated information meta-model.

RG4: To validate the strategic guidelines in relation to *RQ3* with practitioners working in the software-intensive development organizations

Paper V is a validation of the theory of openness presented in *Paper III*. We used a repertory grid technique [95] to analyze and validate the theory of openness. The results showed that both case companies qualify as *laggards* in relation to the theory of openness and neither of them has internal procedures to facilitate developers to contribute to OSS tools communities.

The lack of central tool coordination leads to multiple variants of the same tools, causing additional costs to glue tools together. An important implication for both companies is that they may learn from Sony Mobile's transition from closed tools to open OSS tools by innovating their process in terms of creating a legal framework. Furthermore, both companies can create an internal champion which serves as an interface between the legal department, software developers and top management, to drive the open tools strategy. The framework will help companies to engage their developers in OSS tools communities together with the legal team to facilitate their core product development. Hence, both companies need a centralized, proactive strategy to help software developers use open OSS tools to reduce integration cost.

6 Ethical aspects and threats to validity

Ethical aspects must be taken into consideration in any empirical research activity which involves human subjects or the data related to humans [159]. Singer and Vinson [167] initiated the discussion on ethical issues in software engineering and provide guidelines for the conduct of empirical studies. These guidelines include informed consent, confidentiality of the data from human subjects and weighing the risks, harms and benefits, not only for the individual subjects, but also for the organizations.

OI research in this thesis involves software engineers working in the industry. The investigation started from mining the OSS code repositories to identify the key contributors and classify their contributions in terms of new features, bug fixes, cosmetic issues or documentation. After identifying the key contributors, interviews were conducted with them. It is worth mentioning that the case companies have shown a strong interest in investigating its OI activities to see whether or not it is helping them to accelerate their internal innovation process. For example, Sony Mobile gets recommendation whether or not opening up in their development process gives them a cutting edge on their competitors. Moreover, the researchers are able to publish research papers to carry forward OI state of the art in software engineering. Therefore, it's 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 pick 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. Therefore, the confidentiality of the data collected from the companies is ensured by signing the non disclosure agreements.

Additionally, workshops were conducted in two automotive companies which involves software engineering and managers. All participants were asked to sign a consent document to ensure the voluntary participation of participants. Moreover, the data gathered through these workshops are kept confidential.

Apart from ethical aspects, there are validity concerns worth mentioning about the thesis. Internal validity is the confidence that we can place in the cause and effect relationship in a scientific study [159]. 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 the ability to generalize the study findings [159]. In particular, all those software-intensive organizations using OSS tools in their internal product development. This thesis used Sony Mobile, software companies in the survey and the two case companies from the automotive industry to achieve better external validity of the research work.

Construct validity refers to what extent the studied concepts really represent what the researcher has in mind and what is investigated according to the research questions [159]. Constructs and elements in the theory of openness are derived from literature. However, neither of the case companies come from a software background but they are becoming more and more software-intensive in the development of their core products. Therefore, both companies do not have a well-defined procedure to map all the constructs of the theory. This threat was partially met by keeping the discussion on a higher level to the company's specific context. Furthermore, more software-intensive companies are required to validate the theory of openness.

Reliability deals with the ability to replicate the same study with the same results [159]. 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

Future work may be the extension of *RG4*, which involves further validation in more organizations to extend the generalization of the theory of openness. Furthermore, develop a tool (a web survey), which helps companies conducting a self-assessment with respect to the theory. The aim is to assess the current tool chain of a software-intensive product development organizations. The survey is based on the criteria defined in the theory, and the web tool collects that data and feeds a summary back to the company for their internal use, about their performance in relation to the theory and other companies.

8 Conclusion and main contributions

Even though software engineering tools are not the direct source of revenues, software-intensive organizations rely on these tools for the development of core products. OSS tools (e.g., Jenkins, Gerrit and Git) offer companies an alternate solution to closed source proprietary tools. The OSS tools provide an organization with several benefits as opposed to closed source tools. These benefits may entail free-up developers time, faster development speed, reduced development cost, increased flexibility in tool usage and adaption and govern the open tools ecosystem. However, it must be mentioned that the usage of open tools is not entirely for free if companies want to gain control and steer communities towards their own business model.

Empirical-based insight were provided into this thesis by summarizing the existing evidence on the use of OI by exploiting OSS tools communities. To further strengthen the existing evidence, the case study at Sony mobile helped us understand that software-intensive organizations need proactive management strategies to achieve the standardization of open tools in the long run. Furthermore, the survey in OSS tools communities also helped us understand that software-intensive organizations are keen on using and contributing to these OSS tools communities. However, the empirical evidence suggests a clear lack of guidelines for managers how to engage themselves in the OSS tools. This thesis presents the theory of openness as a main contribution to address the identified research gap.

Theory of openness is an empirically developed theory intended to provide guidelines and helps organizations to utilize OSS tools communities in relation to reduced development cost, shorter time-to-market and process and product innovation.

CAP model provides operational guidelines for software organizations regarding what to conceal and what to share in OSS ecosystems. The model proposes contribution strategies and meta-model to help organization operationalizing these strategies. The goal is to help maximize return on investment and sustain the needed influence on OSS ecosystems.

Validation study validates the theory of openness for software engineering tools in two automotive companies.

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 [6]. Two decades after Allen's paper from 1983, Chesbrough's seminal book about OI [31] has initiated an unabated interest [67] among researchers in innovation management [83], economics, psychology, sociology, and also Software Engineering (SE) [181]. The work initiated by Chesbrough [31] forced both practitioners and scholars to rethink the design of innovation strategies in a networked environment [83]. 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 [65]. More courageous voices suggested even that closed innovation might have been the exception in the history, characterized mostly by open innovation practices [133].

OI is a relatively new field of research and a collective theoretical foundation is starting to emerge. Chesbrough [31] 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 [66], 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 [83], although not the only one [196]. 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 [31]. Similarly in the early days of OSS, many interesting OI initiatives were performed, e.g. opening up software product organizations and utilizing open configurations [88]. 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 [83, 196], partly systematic [186] or, for example, focus on the history or evolution of OSS or available innovation metrics [50]. 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 [85], 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 [147] 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 [85].

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 [100].

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 [186], we identified four secondary studies (literature reviews) on OI [50, 83, 186, 196], relevant to this study. The studies are summarized in Table 1.

Are the reviews systematic? Huizingh [83] and Wnuk and Runeson [196] conducted reviews on OI, however neither of them is systematic according to the guidelines stated by Kitchenham et al. [99]. The study conducted by West and Bogers [186] 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. [50] adheres to guidelines by Kitchenham et al. [99] and Petersen at al. [147]. In this paper, we report a review conducted according to the guidelines by Kitchenham et al. [99].

What were the objectives behind conducting reviews? West and Bogers [186] 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. [54]. Wnuk and Runeson [196] performed a study with the goal to propose a SE framework for OI.

Huizingh [83] 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. [50] 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 [83] neither specified the database, nor the search terms used. Likewise, West and Bogers [186] 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 [196] 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. [50] 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 [86, 161] 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 [196] nor Huizingh [83] used explicit quality assessment criteria for the identified studies. On the other hand, West and Bogers [186] included studies that focused on OI as per the definition by Chesbrough [31] and excluded book reviews, commentaries and editorial introductions. Edison et al. [50] 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. [85] 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 [83, 186, 196]. The information about the mapping between the data extraction properties and the research questions was also absent. However, Edison at al. [50] 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 [37, 38]. Instead, West and Bogers [186] 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. [54]. Similarly, Wnuk and Runeson [196] 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. [193]. 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. [50] 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 [83] 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. [99, 147] and focusing on systematic synthesis of the findings.

Facets	West and Bogers [186] (2013)	Edison et al. [50] (2013)	Wnuk et al. [196] (2013)	Huizingh [83] (2010)
Data sources	1. Academy of Management Review 2. Administrative Science Quarterly 3. California Management Review 4. Harvard Business Review 5. IEEE Transactions on Engineering Mgt. 6. Industrial and Corporate Change 7. Journal of Technology Mgt. 8. Journal of Product Innovation Mgt. 9. Long Range Planning 10. Management Science 11. MIT Sloan Management Review 12. Organization Science 13. R&D Management 14. Research Policy 15. Research-Technology Management 16. Strategic Management Journal 17. 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 [186] (2013)	Edison et al. [50] (2013)	Wnuk et al. [196] (2013)	Huizingh [83] (2010)
Systematic	Partly	Yes	No	No
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	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 SE framework, designed to foster open innovation by designing and tailoring appropriate SE methods and tools.	To explore the challenges faced by practitioners and academics in understanding the OI.

Facets	West and Bogers [186] (2013)	Edison et al. [50] (2013)	Wnuk et al. [196] (2013)	Huizingh [83] (2010)
Outcome	The paper defined the agenda for OI 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 review followed by an online questionnaire and interviews with practitioners and academics to identify the definition of innovation in software industry. Based on the 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. [99] and Petersen et al. [147]. 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 [45] from forward snowball sampling [72, 86].

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.

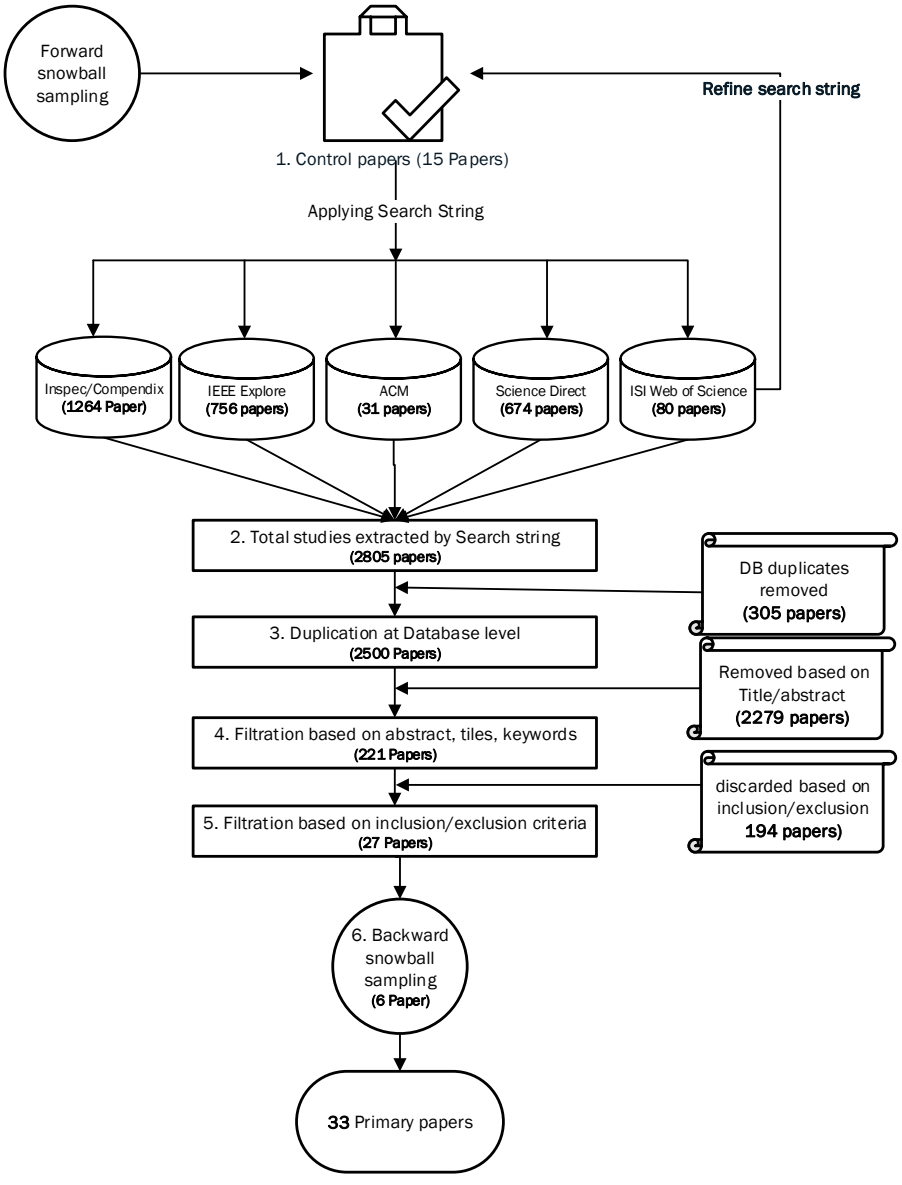
4. Selection of studies based on abstract, titles and keywords: 2279 papers were not found relevant and excluded.

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 [18] 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 [18]. The fact that two of the control papers were not captured by the final search string confirms the observations by Wohlin et al. [197] 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 [146]. Kappa statistics [105] 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 [146], 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. [38]. 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. [85]. 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 2.1 and 2.2. The idea behind investigating rigor and relevance resembles the use of a rubric based evaluation in education [85]. Previous studies [93,132] 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"* [85]. Ivarsson et al. [85] 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 2.1.

Relevance deals with the impact of a study on industry [85]. It consists of manifold aspects, namely, relevance of the topic studied [170], 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 [200], use of research methods that facilitate industrial realism [171], and provision of a realistic situation in terms of users, scale, and context [85]. We followed the suggestion of Ivarsson et al. [85] to decompose rigor into: users/subjects (U), scale (S), research methodology (RM), and context (C), see Appendix 2.2.

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 [159].

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 [147] and the guidelines for quality assessment criteria [85].

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 [31].

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

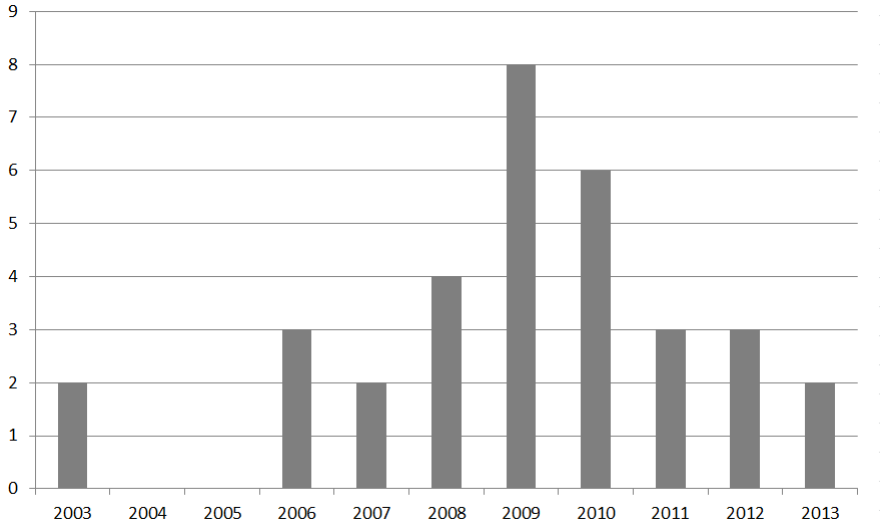
Reliability is concerned with to what extent the data and the analysis are dependent on a specific researcher. Multiple strategies were taken into account in order to enhance reliability. First, there is always a risk of missing out on primary studies with a single search string for all selected databases. 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. [85] 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.

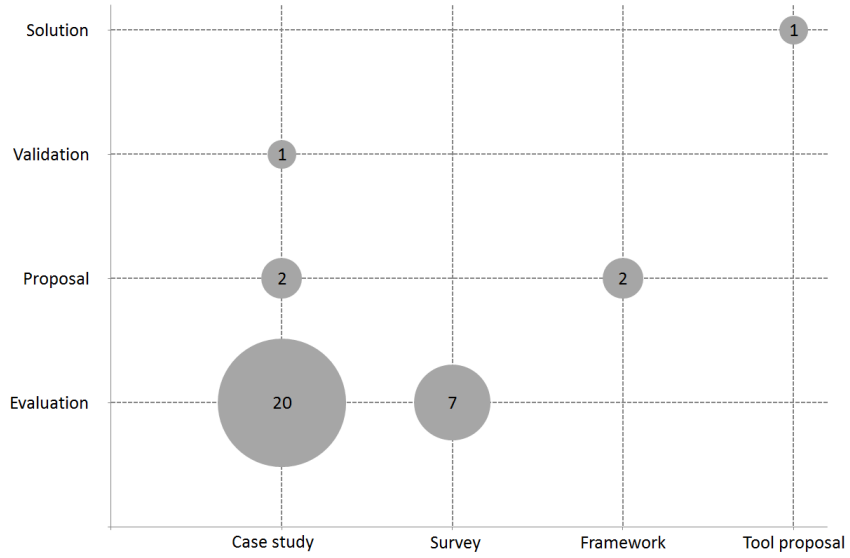
4.1 Distribution of OI studies

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 [83].

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. [159] and vertical axis represents the classification of studies established by Wieringa et al. [193]. 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. [159, 193]



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.

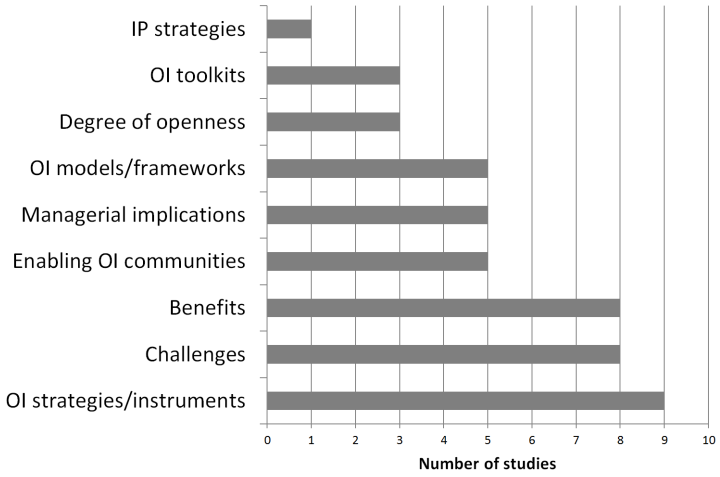
4.3 Thematic analysis

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. [37, 38], 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:

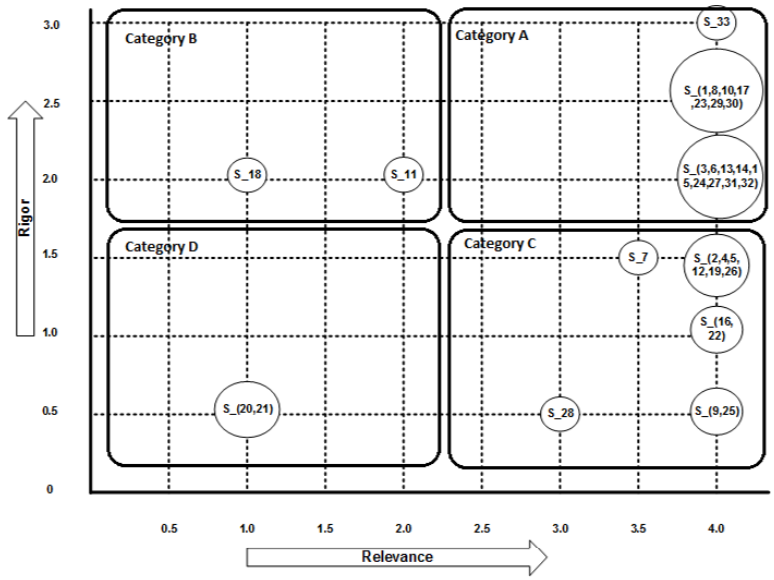
1. Studies with the score from (0–1.5) are considered as low rigor, while high rigor is defined for a score of 2 or above.

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 business

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 [33]. 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 [82]. 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. [100], 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.

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 [85]. 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 [85]. 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	**[42]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_2	**[183]**	0.5	0.5	0.5	**1.5**	1	1	1	1	**4**	C
S_3	**[74]**	1	1	0	**2**	1	1	1	1	**4**	A
S_4	**[35]**	1	0.5	0	**1.5**	1	1	1	1	**4**	C
S_5	**[187]**	1	0.5	0	**1.5**	1	1	1	1	**4**	C
S_6	**[77]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_7	**[49]**	0.5	0.5	0.5	**1.5**	0.5	1	1	1	**3.5**	C
S_8	**[182]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_9	**[190]**	0.5	0	0	**0.5**	1	1	1	1	**4**	C
S_10	**[145]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_11	**[196]**	1	0.5	0.5	**2**	0	0	1	1	**2**	B
S_12	**[130]**	0.5	0.5	0.5	**1.5**	1	1	1	1	**4**	C
S_13	**[176]**	1	1	0	**2**	1	1	1	1	**4**	A
S_14	**[194]**	0.5	0.5	1	**2**	1	1	1	1	**4**	A
S_15	**[40]**	1	1	0	**2**	1	1	1	1	**4**	A
S_16	**[185]**	0.5	0.5	0	**1**	1	1	1	1	**4**	C
S_17	**[13]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_18	**[153]**	0.5	1	0.5	**2**	0	1	0	0	**1**	B
S_19	**[47]**	0.5	0.5	0.5	**1.5**	1	1	1	1	**4**	C
S_20	**[168]**	0	0.5	0	**0.5**	0	0	1	0	**1**	D
S_21	**[84]**	0	0.5	0	**0.5**	0	0	0	1	**1**	D
S_22	**[179]**	0.5	0.5	0	**1**	1	1	1	1	**4**	C
S_23	**[34]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_24	**[131]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_25	**[156]**	0.5	0	0	**0.5**	1	1	1	1	**4**	C
S_26	**[76]**	1	0.5	0	**1.5**	1	1	1	1	**4**	C
S_27	**[88]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_28	**[24]**	0	0.5	0	**0.5**	1	1	0	1	**3**	C
S_29	**[73]**	1	0.5	1	**2.5**	1	1	1	1	**4**	A
S_30	**[46]**	1	1	0.5	**2.5**	1	1	1	1	**4**	A
S_31	**[78]**	1	0.5	0.5	**2**	1	1	1	1	**4**	A
S_32	**[106]**	1	1	0	**2**	1	1	1	1	**4**	A
S_33	**[173]**	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

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

Abstract

Background. Despite growing interest of Open Innovation (OI) in Software Engineering (SE), little is known about what triggers software organizations to adopt it and how this affects SE practices. OI can be realized in numerous of ways, including Open Source Software (OSS) involvement. Outcomes from OI are not restricted to product innovation but also include process innovation, e.g. improved SE practices and methods.
Aim. This study explores the involvement of a software organization (Sony Mobile) in OSS communities from an OI perspective and what SE practices (requirements engineering and testing) have been adapted in relation to OI. It also highlights the innovative outcomes resulting from OI.
Method. An exploratory embedded case study investigates how Sony Mobile use and contribute to Jenkins and Gerrit; the two central OSS tools in their continuous integration tool chain. Quantitative analysis was performed on change log data from source code repositories in order to identify the top contributors and triangulated with the results from five semi-structured interviews to explore the nature of the commits.
Results. 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 organization. ii) Assets not seen as competitive advantage

nor a source of revenue are made open to OSS communities, and gradually, the organization turns 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 included free features and maintenance, and were believed to increase speed and quality in development.
Conclusion. Adopting OI was a result of a paradigm shift of moving from Windows to Linux. This shift enabled Sony Mobile to utilize the Jenkins and Gerrit communities to make their internal development process better for its software developers and testers.

1 Introduction

Software organizations have recently been exposed to new facets of openness that go beyond their experience and provide opportunities outside their organizational walls. Chesbrough [31] explains the term *Open Innovation* (OI) as *"a paradigm that assumes that organizations 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 is based on *outside-in* and *inside-out* knowledge flows that help to advance technology and spark innovation. Some classical examples of inside-out are selling intellectual property while outside-in correspond to start-up acquisition and integration. There are also *coupled processes* [54] where companies give and take during co-creation by making alliances and joint-ventures. OI is fuelled by increased mobility of workers and knowledge, more capable universities, greater knowledge access and sharing capabilities that World Wide Web offers [30] and easier access to venture capital for start-ups.

Open Source Software (OSS) was widely used by software organizations before the OI model became popular [112] and nowadays provides a common example of OI [141]. OSS leverages external resources and knowledge to increase innovation, product quality and to shorter time-to-market. OSS offers not only potential product innovation (e.g. by using an OSS platform of commodity parts to build differentiation parts), but potential process innovations in terms of an implementation of new or significantly improved production or delivery methods [117].

IBM's engagement in the Linux community in terms of patent and monetary contributions exemplifies how a firm can leverage OSS from an OI perspective. Risks and costs of development were in this case shared among other stakeholders such as Intel, Nokia, and Hitachi, which also have made significant investments in the Linux community [110]. Thanks to Linux involvement, IBM can strengthen 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 customers [30].

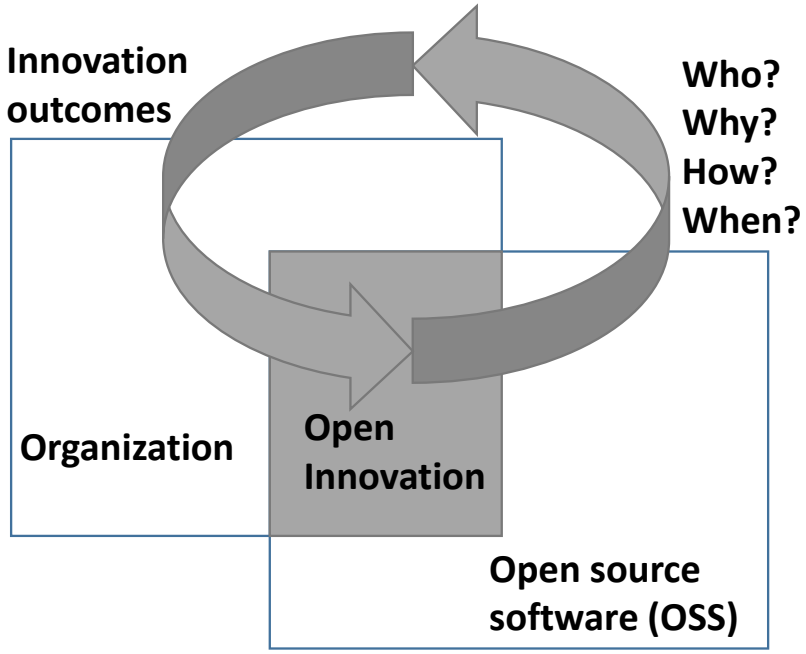


Figure 1: Study Objectives in the intersection between proprietary organizations and open source software.

Software organizations that want to benefit from OI via OSS engagement need to adapt and innovate their internal software development strategies and processes. For example, influence on feature selection and road-mapping may be gained through a more active participation, as many OSS communities are based on meritocracy principles [91]. Also, some benefits may first be fully utilized after contributing back certain parts to the OSS community [180]. For example, by correcting bugs, actively participating in discussions and contributing new features, a software organization might reduce maintenance cost compared to proprietary software development [176]. Hence, in order for a firm to gain the expected benefits of products, OI process innovations may be a required step on the way forward [108, 157, 194]. Existing literature does not particularly focus on how these internal SE process adaptions should be structured or executed [141]. Further, little is known about how OSS involvement may be utilized as an enabler and support for further innovation spread inside an organization, e.g. process, tools, or organizational innovations.

In this study, we focus on identifying when, why and how a software organization adopts OI through the use of OSS, and what innovative outcomes can be

gained (see Fig. 1). We investigate these aspects through a case study at Sony Mobile and how it actively participate and contribute to the communities of the two OSS tools Jenkins and Gerrit. These two tools are the basis of Sony Mobile's internal continuous integration tool chain. The study further investigates how external knowledge and innovation captured through the active development of these OSS tools may be transferred into the product development teams of Sony Mobile. More explicitly, this study contributes by studying how OSS may be used, not only for leveraging product innovation [117] 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 product quality.

This paper is structured as follows. Section 2 highlights the related work and Section 3 outlines the research methodology. In Sections 4 and 5 results from the quantitative and qualitative analysis are presented, respectively. Finally, Section 6 discussed the results, followed by conclusions in Section 5.

2 Related work

In this section, we summarize related work in OI strategies, OI challenges in SE and open source development practices inside software organizations. This section is partly based on the systematic mapping study by Munir et al. [138].

The increased openness that OI implies poses significant challenges to software organizations in terms of securing their competitive advantage [141] and understanding what to contribute, when and how to maintain differentiation towards competitors that may also be involved in the OSS community [76, 89, 120]. Related to that is the challenge of what requirements should be selected, when these should be released and how an internal roadmap should be synchronized with the OSS project's roadmap [119, 194]. These challenges highlight the need for a clear contribution strategy that software organizations should create to focus their internal resources on value-creating activities, rather than contributing unnecessary patches or differentiating features [194].

Extensive involvement in OSS communities may also bring significant challanges. Among these challenges, Daniel et al. [44] suggested that the conflict between organizational and OSS standards reduces developers' organizational commitment and it is strongly dependent on the degree to which developers associate themselves with organizations or OSS communities. Investing in OSS may also be costly and create differentiation and property right protection challenges, as indicated by Stuermer et al. [176] who studied the Nokia Internet Tablet, which was based on a hybrid of OSS and proprietary software development.

West et al. [190] examined the complex ecosystem surrounding Symbian Ltd. and identified three inherent difficulties for organizations 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.

Looking at OI strategies, Dahlander & Magnusson [40] show how organizations may access OSS communities in order to extend the firm's resource base, align the organization's strategy with that of the OSS community, and/or assimilate the community in order to integrate and share results with them. The same authors explained that depending on how open a firm chooses to be in regards to their business model, different strategies may be enforced, e.g. symbiotically giving back result to the community, or as a free-rider keeping modifications and new functionality to oneself [41]. Some strategies include:

- selectively revealing - differentiating parts are kept internal while commodity parts are made open [76, 185]. This requires continuous assessment of what parts are to be considered commodity as opposed to differentiating value.
- licensing schemas (cf. Dual-licensing [32]), technology may be fully disclosed, but under a restrictive license [185]. Alternatively, everything may be disclosed under open and transparent conditions [32].

Henkel [76] reports how small organizations reveal more, as they are likely to benefit from the external development support. Component manufacturers also reported to contribute a lot as they have a good protection of the hardware they sell; software is seen as a complementary asset. In a follow-up study, Henkel [78] further reported how openness had become a competitive edge, as customers had started to request even more revealing.

Dahlander & Wallin [42] show how having an employee in the community can be an enabler for the organizations to not only gain a good reputation but also to influence the direction of the development towards the organizations' own interests. However, to gain the roles needed to commit or review code written by community developers, individuals need to contribute and become an active part of the communities as these are often based on the principles of meritocracy [91].

Inner Source [174] has gained interest among researchers and practitioners as a way to adapt OSS practices at software organizations. Such hybrids of commercial and OSS practices [127] could include using the OSS style project structure, where a core team of recognized experts has the power to commit code to an official release, and a much larger group contributes voluntarily in many ways.

Summary. Research has shown a lot of interest for OI and its different applications [186], including leveraging OSS for OI [141]. However, the focus is mostly limited to management and strategic aspects, e.g., [40, 176, 191], with some exception of inner sourcing [130, 174]. Little is still known about what triggers software organizations to adopt OSS from an OI perspective and how this affects SE practices [141].

This paper adds to existing knowledge by focusing on the use of OSS from an OI perspective in an organization that seek to complement its internal product de-

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 OI?	To explore the transition from a closed innovation process to an OI process.
RQ3: How does Sony Mobile take a decision to make a project or feature open source?	To investigate what factors affect the decision process when determining whether or not 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 the OI adoption?	To investigate the requirements engineering and testing processes and how they deal with the special complexities and challenges involved due to OI.

velopment and process innovation [117] with the use of external knowledge from OSS communities. Furthermore, this study aims to improve our understanding of what and how a software organization can open up and how SE practices are adapted to deal with the openness to OSS communities.

3 Case study design

Below we describe the research design of this study. We explain the research questions, the structure of the case study design, and the methodologies used for data collection as well as for the quantitative and qualitative analysis.

3.1 Research questions

The focus of this study is on how software organizations use OSS projects from an OI perspective, what triggers them to open up and how this impacts the organiza-

tions' innovative performance and their SE practices (see Fig. 2). We investigate these aspects through a case study at Sony Mobile, and how they actively participate and contribute to the communities of the two OSS tools Jenkins [2] and Gerrit [3]. Both tools constitute pivotal parts in Sony Mobile's internal continuous integration tool chain.

The study further investigates how external knowledge and innovation captured through the development of these OSS tools, may be transferred into the product development teams of Sony Mobile. More explicitly, this study contributes by studying how OSS may be used, not only for leveraging product innovation [117] 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 within the organization.

1. ***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.

2. ***Gerrit code review*** is an OSS code review tool created by Google in connection with the Android project 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 it is allowed to be merged into the main branch.

Based on this background, and the research gap identified in earlier work [141], we formulate our research questions to study the OI in Sony Mobile in an exploratory manner (see Table 1). *RQ1* addresses the extent to which Sony Mobile is involved in the Jenkins and Gerrit communities 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. *RQ4* highlights the key innovation outcomes realized as a result of openness. Finally, *RQ5* aims at understanding whether or not the existing requirements engineering and testing processes have the capacity to deal with the OI challenges in SE. *RQ1* is answered with the help of quantitative analysis of repository data, while the remaining four research questions (*RQ2, RQ3, RQ4, RQ5*) are 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, developing embedded devices. The studied branch focuses on developing Android-based phones and tablets and has 1600 employees, of which 900 are directly involved in software development. Sony Mobile develops software in an agile fash-

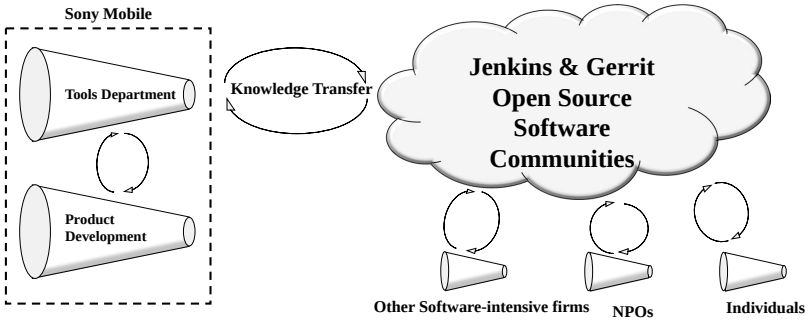


Figure 2: The Jenkins and Gerrit OSS communities surrounded by Sony Mobile and other members. Arrows represent knowledge transfer in and out of the community members such as other software organizations, non profit organizations (NPO) and individuals, which in turn are illustrated by funnels, commonly used in OI literature [31].

ion and applies software product line management with a database of more than 20,000 features suggested or implemented across all product lines [150].

However, in order to work with OSS communities, namely Jenkins and Gerrit Sony Mobile created a designated tools department to acquire and integrate the external knowledge to improve the internal continuous integration process. The continuous integration tool chain used by Sony Mobile is developed, maintained and supported by an internal tools department. The teams working on phones and tablets are thereby relieved of this technical overhead. During the recent years, Sony Mobile has transitioned from passive usage of the Android codebase into active involvement and community contribution with many code commits to Jenkins and Gerrit. This maturity resulted in a transition from closed innovation to OI [31], assuming that business values are created or captured as an effect.

From an OI perspective, there are interactions between the Tools department and the Jenkins and Gerrit communities (see Fig. 2). The in- and outgoing transactions, visualized by the arrows in Fig. 2, are data and information flows, e.g. ideas, support and commits, can be termed as a coupled innovation process [54]. 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 Tools department accesses, adapts and integrates the externally obtained knowledge from the Jenkins and Gerrit communities into the product development teams of Sony Mobile.

This creates additional transactions inside Sony Mobile which can be labeled as process innovation [4] in the sense that new tools and ways of working improve development efficiency and quality. This relates to the internal complementary assets need that is mentioned as an area for future research by Chesbrough et al. [29].

We conducted a case study design with Jenkins and Gerrit as units of analysis [159] as these are the products in which the exchange of data and information enable further innovation inside Sony Mobile.

3.3 Case study procedure

We performed the following steps.

1. Preliminary investigation of Jenkins and Gerrit repositories.
2. Mine the identified project repositories.
3. Extract the change log data from the source code repositories.
4. Analyze the change log data (i.e. stakeholders, commits etc).
5. Summarize the findings from the change log data to answer *RQ1*.
6. Prepare and conduct semi-structured interviews to answer *RQ2–RQ5*.
7. Synthesize data.
8. Answer the research questions *RQ1–RQ5*.

3.4 Methods for quantitative analysis

To understand Sony Mobile's involvement in the OSS tools (*RQ1*), we conducted quantitative analysis of commit data in the source code repositories of Jenkins and Gerrit.

Preliminary Investigation of Jenkins and Gerrit Commits

A *commit* is a snapshot of a developer's files after reaching a code base state. The number of lines of code in a commit may vary depending upon the nature of the commit (e.g. new implementation, update etc.) [75]. The comment of a commit refers to a textual message related to the activity that generates the updated new piece of code. It ranges from a simple note to a detailed description, depending on the project's conventions. In this study, we used the keywords provided by Hattori [75] in his study as a reference point to classify the commit messages (see Table 2).

We mined the source code repositories of Jenkins and Gerrit to extract the commit id, date, committer name, committer email and commit description message

for each commit, with the help of the tool CVSAnlY [1]. The extracted data was stored locally in a relational database with a standard data scheme, independent of the analyzed code repository. The structure of the database allows a quantitative analysis to be done by writing SQL queries. The number of commits per committer were added together with the name and email of the committer as keys.

We extracted the affiliations of the committers from their email addresses by filtering them on the domain, e.g., john.doe@sonymobile.com was classified with a Sony Mobile affiliation. It is recognized that committers may not use their corporate email addresses when contributing their work, since parts of their work could be contributed privately or under the umbrella of other organizations than their employer. To triangulate and complement this approach, a number of additional sources were used, as suggested by earlier research [19, 70]. First, social media sites as LinkedIn, Twitter and Facebook were queried with keywords from the committer, such as the name, variations of the username and e-mail domain. Second, unstructured sources such as blogs, community communication (e.g., comment-history, mailing-lists, IRC logs), web articles and firm websites were consulted.

Sony Mobile turned out to be one of the main organizational affiliations among the committers to Gerrit while no evidence of commits to the Jenkins core community was identified. The reason for this was that Jenkins is a plug-in-based community, i.e. there is a core component surrounded by approximately 1,000 plug-ins of which each has a separate source code repository and community. Our initial screening had only covered the core Jenkins component. After analyzing forum postings, blog posts and reviewing previously identified committers, a set of Jenkins plug-ins, as well as two Gerrit plug-ins, were identified, which then were also included in our analysis. The following Open Source projects were included for further analysis:

- Gerrit[1]
- PyGerrit (Gerrit plug-in)[2]
- Gerrit-events (Gerrit plug-in)[3]
- Gerrit-trigger (Jenkins plug-in)[4]
- Build-failure-analyzer (Jenkins plug-in)[5]
- External-resource-viewer (Jenkins plug-in)[6]

[1] https://www.openhub.net/p/gerrit
[2] https://www.openhub.net/p/pygerrit
[3] https://www.openhub.net/p/gerrit-events
[4] https://github.com/jenkinsci/gerrit-trigger-plugin
[5] https://www.openhub.net/p/build-failure-analyzer-plugin
[6] https://github.com/jenkinsci/external-resource-dispatcher-plugin

- Team-views (Jenkins plug-in)[7]

Classification of commit messages

Further analysis included creating the list of top committers combined with their yearly activity (number of commits) in order to see how Sony Mobile's involvement evolved over time. Next, we characterized and classified the commits made by Sony Mobile to the corresponding communities by following the criteria defined by Hattori et al. [75]. This was done manually by analyzing the description messages of the commits and searching for keywords (see Table 2), and then classifying the commits in one of the following categories:

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 of the code without adding new features. **Corrective engineering** activities refer to fixing defects 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 the total commit messages were taken to classify the commit messages and Kappa was calculated to be 0.29. Consequently, disagreement was 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 had laid a foundation to understand the relation between Sony Mobile, and the 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*. Interview questions are listed in the Appendix.

Interviewee selection

The selection of interviewees was based on the committers identified in the initial screening of the projects. Three candidates were identified and contacted by e-mail (Interviewees 1, 2 and 3, see Table 3). Interviewees 4 and 5 were proposed during the initial three interviews. The first three are top committers 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 interviewee I3, when

[7] https://github.com/jenkinsci/team-views-plugin

Table 2: Keywords used to classify commits taken from Hattori [75].

Forward Engineering	**Re-engineering**	**Corrective Engineering**	**Management**
IMPLEMENT	OPTIMIZ	BUG	CLEAN
ADD	ADJUST	ISSUE	LICENSE
REQUEST	UPDATE	ERROR	MERGE
NEW	DELET	CORRECT	RELEASE
TEST	REMOV	PROPER	STRUCTURE
START	CHANG	DEPRAC	INTEGRAT
INCLUD	REFACTOR	BROKE	COPYRIGHT
INITIAL	REPLAC		DOCUMENTATION
INTRODUC	MODIF		MANUAL
CREAT	ENHANCE		JAVADOC
INCREAS	IMPROV		COMMENT
	DESIGN		MIGRAT
	CHANGE		
	RENAM		REPOSITORY
	ELIMINAT		CODE RE-VIEW
	DEUPLICAT		POLISH
	RESTRUCTUR		UPGRADE
	SIMPLIF		STYLE
	OBSOLETE		FORMATTING
	REARRANG		ORGANIZ
	MISS		TODO
	ENHANCE		
	IMPROV		

he was contacted, had just left Sony Mobile for a smaller organization dedicated to Jenkins development. His responsibilities as the tools manager for Jenkins at Sony Mobile were taken over by interviewee I4. Interviewee 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 on the tools. Interviewee 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.

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 explore other relevant findings during the interview wherever needed. The two first authors were present during all five interviews, with the

Table 3: Interviewee demographics.

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

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. The duration of the interviews varied from 45 to 50 minutes.

3.6 Validity threats

This section highlights the validity threats related to the case study. Four types of validity threats [159] are addressed with their mitigation strategies.

Internal validity

This concerns causal relationships and the introduction of potential confounding factors.

Confounding factors. To mitigate the risk of introducing confounding factors, the study was performed on the tools level instead of an organizational level to ensure that the innovation outcomes are merely the result of adopting OI. Performing the study on an organization level introduces the risk of confounding the

innovation outcomes as a result of a product promotion or financial investment etc. instead of the use of external knowledge from OSS communities. Therefore, a more fine-grained analysis on the OSS tools level was chosen to minimize the threat of introducing confounding factors.

Subjectivity. It was found in the study that Sony Mobile does not use any general innovation metrics to measure the impact of OI. Therefore, researchers had to rely on qualitative data. This leads to the risk of introducing subjectivity while inferring innovation outcomes as a result of OI adoption. In order to minimize this risk, the first two authors independently performed the analysis and the remaining authors reviewed it to make the synthesis more objective. Moreover, findings were sent back to interviewees for validation. Furthermore, subjectivity was minimized by applying the commit messages classification criteria proposed by Hattori et al. [75]. During the analysis, the disagreements were identified using Kappa analysis and resolved to achieve a substantial agreement.

Triangulation. In order to mitigate the risk of identifying the wrong innovation outcomes, we used multiple data sources by mining the Jenkins and Gerrit source code repositories prior to conducting interviews. Furthermore, we also performed observer triangulation during the whole course of the study to mitigate the risk of introducing researcher bias.

External validity

This refers to the extent it is possible to generalize the study findings to other contexts. The scope of this study is limited to a software organization utilizing the notion of OI to accelerate its innovation process. The selected case organization 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 organizations making Android phones. The involvements by other 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 commits to Jenkins and Gerrit, and other OSS tools used in the tool chain development. Our findings may also be relevant to software organizations with similar context, domain and size as Sony Mobile.

Construct validity

This refers to what extent the operational measures that are studied really represent what researcher has in mind, and what is investigated according to the research questions [159]. We took the following actions to minimize construct validity threats.

Selection of interviewees. We conducted a preliminary quantitative analysis of the Jenkins and Gerrit repositories to identify the top committers and to select

the relevant interviewees. The selection was performed based on the individuals' commits to Jenkins or Gerrit. Moreover, recommendations were taken from interviewees for suitable further candidates to attain the required information on OI. Process knowledge, role, and visible presence in the community were the key selection factors.

Reactive bias. Researchers presence might limit or influence the interviewees and causing them to hide facts or respond after assumed expectations. This threat was limited by the presence of a researcher that has a long research collaboration record with Sony Mobile and explained confidentiality rules. Furthermore, interviewees were ensured anonymity both within the organization and externally in the OSS community.

Design of the interviews. All authors validated the interview questionnaire followed by a pilot interview with an OSS Jenkins community member in order to avoid misinterpretation of the interview questions.

Reliability

The reliability deals with to what extent the data and the analysis are dependent on the specific researcher, and the ability to replicate the study.

Member checking. To mitigate this risk, multiple researchers individually transcribed and analyzed the interviews to make the findings more reliable. In addition, multiple data sources (qualitative and quantitative) were considered to ensure the correctness of the findings and cross-validate them. All interviews were recorded, transcribed and sent back to interviewees for validation. The commit database analysis was performed and validated by multiple researchers.

Audit trail. Researchers kept track of all the mined data from OSS code repositories as well as interview transcripts in a systematic way to go back for validation if required. Finally, this study was not ordered by Sony Mobile to bring supporting evidence for OI adoption. Instead the idea was to keep the study design and findings as transparent as possible without making any adjustments in the data except for the anonymizing the interviewees. The results were shared with Sony Mobile prior to submitting the study for publication.

4 Quantitative analysis

This section presents a quantitative analysis of commits made to eight OSS projects, namely: Gerrit, pyGerrit, Gerrit-events, Gerrit-trigger, Build-failure-analyzer, External-resource-viewer and Team-views as depicted in section 3.4. It should be noted that the seven latter projects are plugins to Gerrit and Jenkins, i.e., not part of the core projects. In the analysis we investigated the types of commits made (see Section 3.4), and in what proportion these were made by Sony Mobile over time, as well as compared to other major organizations.

Commits classification	2010	2011	2012	2013	2014	Total
Forward Engineering	65	44	264	373	207	953
Re-engineering	38	65	240	336	190	869
Corrective engineering	10	12	59	62	26	169
Management	12	15	96	171	73	367
Total	125	136	659	942	496	2358

Table 4: Sony Mobile's commits to Gerrit analyzed per year.

4.1 Gerrit

The two largest categories of commits for Gerrit are forward engineering (953 commits) and re-engineering (869 commits), followed by management commits (367 commits) and corrective engineering commits (169 commits), see Table 5.

Table 5: Classification of Sony Mobile's commits to OSS tools based on hattori's criteria [75]

Tools	Forward Engineering	Re-Engineering	Corrective Engineering	Management
Gerrit	953	869	169	367
pyGerrit	27	18	19	36
Gerrit-events	27	18	19	36
Gerrit-trigger	60	40	76	135
Build-failure-analyzer	60	19	17	36
External-resource-viewer	28	8	8	6
Team-views	7	0	0	5

This dominance of forward and re-engineering commits remained stable between 2010 and 2014, see Table 4. Sony Mobile presented the first Android-based mobile phone in March 2010 and as can be seen from the analysis also became active in contributions to Gerrit with a total of 125 contributions in 2010. From 2012 the number of forward and re-engineering commits became more equal each year suggesting that Sony Mobile was not only contributing new features but also actively helping in increasing the quality of the current features and re-factoring.

The number of forward engineering and re-engineering commits remained high and we notice a substantial decrease of corrective engineering and management commits. The decrease of management commits may suggest that Sony Mobile reached a high level of compatibility of its code review processes and therefore requires fewer commits in this area. This data shows an interesting pattern in joining an OSS community. Since Sony Mobile is a large organization with several complex processes, their joining of the Gerrit community had to be associated with a substantial number of forward engineering and re-engineering commits. This entry to the community lowered the transition time and enabled faster synchronization of the code review processes between the Android community players and Sony Mobile. At the same time, Sony Mobile contributed several substantial features from the first year of participation which is positive for the community. Figure 3 shows the progression of commits made by Sony Mobile to all OSS tools between year 2009 and 2014.

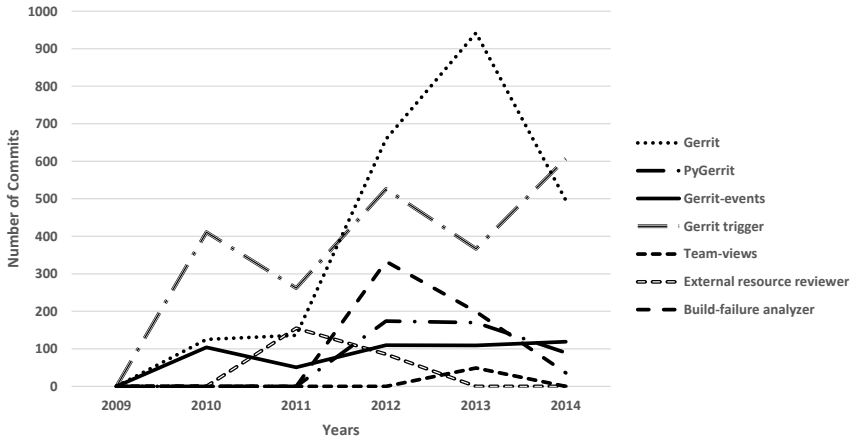


Figure 3: Sony Mobile's commits for all OSS tools per year

PyGerrit

PyGerrit is a Python library that provides a way for clients to interact with Gerrit. As can be seen in Table 6, Sony Mobile initiated this plug-in and is the biggest committer to it, representing 97.5% of the commits. Management commits are the most frequent category, followed by forward engineering commits. This suggests that some code formatting changes, cleaning up code and documentation commits were delivered by Sony Mobile after opening up this plug-in to the community. Sony Mobile's yearly contribution analysis shows a steady growth since its introduction in 2011 (see Fig. 3).

Table 6: Percentage of Sony Mobile's contribution compared to other Software organizations

Tools	Sony	Google	Ericsson	HP	SAP	Intel	Others
Gerrit	8.2	38.5	0	0	10.7	0	42.5
PyGerrit	97.5	0	0	0	0	0	2.4
Gerrit-event	66.1	0	3.3	4.1	0.2	2	24.2
Gerrit-trigger	65.2	0	9.1	2.4	0.7	1.3	21.2
Team-views	100	0	0	0	0	0	0
External-resource-reviewer	89.6	1.5	4.8	0	0	0	4.1
Build-failure-analyzer	85.5	0	0	0	0	0	14.4

Conclusion: This indicates that companies that want the communities to accept their plug-ins should be prepared to dedicate effort on management type of commits to increase the code's quality and documentation and therefore enable other players to contribute.

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. Table 6 shows that apart from Sony Mobile(66.1%), HP(4.1%), SAP(0.2%), Ericsson(3.3%) and Intel(2%) commits 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 committer along with its competitors (see Table 6). Similarly, to the PyGerrit plug-in, management and forward engineering commits dominate and Sony Mobile is the main driver of features to this community.

Conclusion: Sony Mobile turns out to be the biggest contributor in Gerrit-event where the focus is mostly on adding new features (forward engineering) based on the internal organizational needs.

4.2 Jenkins

Commits from Sony Mobile to Jenkins could not be identified in the core product but to a various set of plug-ins (see Table 6). The ones identified are:

- Gerrit-trigger
- Build-failure-analyzer
- External resource-reviewer
- Team-views

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 Table 6) seems to attract the most number of commits with the percentage of 65.2% from Sony Mobile. 135 commits were classified as management and 76 as corrective engineering. In this case, the majority of the commits were not forward or re-engineering, which may suggest that Sony Mobile was more interested in increasing the code quality and fixing the bugs rather than extending it. It seems logical as for the Jenkins community new functionality can be realized in the form of a new plug-in rather than extending the current plug-ins.

Conclusion: Adding plug-ins allows greater flexibility but increases the total number of parallel projects to manage and maintain by the community.

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 see Table 6, Sony Mobile came out as the largest committer (85.5%) to the Build-failure-analyzer. One possible explanation for the lack of contribution from the other software organizations is that this plug-in might be very specific to the needs of Sony Mobile, but they made it open to see if the community shows interest in contributing to further development efforts.

Forward engineering and management commits are the two most common categories. Moreover, the number of commits have declined after 2012 and Table 5 shows a relatively low numbers of corrective engineering (17) and re-engineering (19) commits, which seem to indicate the maturity of this plug-in in terms of quality and functionality.

Conclusion: We hypothesize that after creating and contributing the core functionality for a given plug-in, the number of forward commits declines and further advances are realized in a form of a new plug-in.

External-resource-viewer

This plug-in adds support for external resources in Jenkins. An external resource is something attached to a Jenkins slave and can be locked by a build, to get exclusive access to it, then released after the build is done. Examples of external resources are phones, printers and USB devices. Like Build-failure-analyzer, Sony Mobile's is the top commiter with the largest contribution percentage of 89.6% compared to Google (1.48%) and Ericsson (4.8%). Moreover, the majority of the commits are classified as forward engineering, suggesting that Sony Mobile has come up with the majority of the functionality to this plug-in. As the number of corrective engineering and re-engineering commits remained low (8 commits in each category), we can assume that the contributed code was high quality.

Conclusion: This data suggest a hypothesis that companies that frequently interact with OSS communities learn to contribute high quality code and possibly keep the same quality standards for other development initiatives.

Team-views

This plug-in provides teams, sharing one Jenkins master, to have their own area with team-specific views. Sony Mobile turned out to be the only committer for this tool (see Table 6), which implies that Team-views is tailored for the needs of Sony Mobile. Only forward engineering and management commits were identified in the data, suggesting that high quality code was contributed and no major refactoring was required for this plug-in. This result also supports our previous hypothesis that modular plug-in based OSS communities provide an efficient way for proprietary companies to participate and contribute with new functionality as new plug-ins.

Conclusion: Decoupling of plug-ins helps in targeting contributions and quality improvement suggestions and simplifies the collaboration networks for discussions on bugs and future improvements.

5 Qualitative analysis

We conducted thematic analysis [37, 38] to find recurring patterns in the collected qualitative data. The following steps were performed in the process.

1. Transcribe the interviewed data from the five interviewee (see Table 3).
2. Identify and define five distinct themes in the data (see Table 7).
3. Classify the interview statements based on the defined themes.
4. Summarize the findings and answers to the RQs.

Table 7: Themes emerging from the thematic analysis.

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

5.1 Opening up

The process of opening up for external collaboration and maturing as an open source organization, can be compared to moving from a closed innovation model to an OI model [29]. 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 [191]. 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 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 active committer in the ways we can control, understand and even steer it up to the way we want to have it"*. The biggest hurdle concerned the notion of giving away internally developed intellectual property rights, which could represent competitive advantage. The legal department needed some time to understanding the benefits and license aspects, which caused the initial contribution process to be extra troublesome. In this case, Sony Mobile benefited from having an internal champion and OSS evangelist (I5). 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 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 and discuss OSS related issues. The OSS contribution process now includes submitting a form for 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. Sony Mobile's involvement in OSS communities is in-line with the findings of governance in OSS communities by Jensen [92].

Conclusion: Adopting OI was a result of a paradigm shift moving from Windows to Linux environment to stay as close as possible to Google's tool chain. Furthermore, Sony Mobile saw a great potential in contributing to OSS communities (Jenkins and Gerrit) and steering them towards its own organizational interests, as opposed to buying costly proprietary tools.

5.2 Determinants of openness

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 source of revenue. 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 less influence and will have to adapt to the governance and requirements from

the others. A good example here is the Gerrit-trigger. The functionality was requested internally at Sony Mobile and therefore undergone development by the Tools department during the same period it became known that there was a similar development ongoing in the community. As stated by I3, *"...we saw a big risk 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"*.

Conclusion: One of the key determinants of making a project open is that it is not seen as a main source of revenue. In other words, there is no competitive advantage gained by Sony Mobile by retaining the project in-house. By maintaining an internal fork, the project incurs more maintenance cost compared to making it open source. Therefore, all the all projects with no competitive advantage are seen as good candidates to become open source.

5.3 Requirements engineering

This theme 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, which is a common characteristic for OSS [163]. The product development teams in Sony Mobile are the main customers of the Tools department. The teams 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 the 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[8] 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

[8]https://wiki.jenkins-ci.org/display/JENKINS/BuildFailureAnalyzer

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 plug-in) 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 it is a good idea [for Sony Mobile]"*. If it is not interesting enough but still a good idea, they are open for commits from the community.

An example regards the Gerrit-trigger plug-in and the implementation of different trigger styles. Pressing issues in the Tools department's 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 a 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 the 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 an organization that 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 *"... it is quite common for these types of collaboration to happen just between plug-in 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 [130]. 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 changes 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"*.

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 webcasts 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 their system and we would loose the whole way of getting help from community to fix stuff and collaborate on issues"*. Another risk would be that *"... all of a sudden everybody is dependent on stuff that is taken away from the 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 [180, 194]. An issue in Jenkins is that there exist numerous combinations and settings of plug-ins. Therefore, it is very important to have backward compatibility when updating a plug-in and planning new features.

Conclusion: The requirements engineering process does not seem to be very rigid, and a majority of the features requests are submitted through e-mails, and monthly workshops with the power users (e.g. internal developers and testers). However, large features are discussed directly with the community at hackathons by the Sony Mobile's Tools department to avoid communication bottlenecks. Furthermore, the prioritization of features is based on the internal needs of Sony Mobile.

5.4 Testing

Similar to 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, *"... Organizations 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 [42]. 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"*.

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 uses Selenium[9], which is an external OSS test framework 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 gathered, including representatives from the Tools de-

[9] http://www.seleniumhq.org/

partment. 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 role in the Selenium Java transition process, *"The idea of an acceptance test harness came from the community but [Sony Mobile] was the biggest committer 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 a new stable version. From the community perspective I3 stated, *"an Acceptance Test Harness also helps the community and other Organizations to understand what problems that big or small organizations have in terms of features or in terms other requirements on the system. So it's a tool where everyone helps each other"*.

Conclusion: Like the requirements engineering process, the testing process is also very informal, and Sony Mobile prioritizes the issues that affect them the most. One of the biggest challenges faced by the community and organizations is to have complete test coverage due to the open plug-in architecture. The introduction of an acceptance test harness was an important step to make the whole testing process automated for organizations, and the Jenkins and Gerrit communities.

5.5 Innovation outcomes

The word *innovation* has a connotation of newness [8] and can be classified as either things (products and services), or changes in the way we create and deliver products, services and processes. Assink [8] classified innovation into disruptive and incremental. Disruptive innovations change the game by attacking an existing business and offering great opportunities for new profits and growth. Incremental innovations remain within the boundaries of the existing technology, market and technology of an organization. The innovation outcomes found in this study are related to incremental innovations.

Sony Mobile does not have any metrics for measuring process and product innovation outcomes. However, valuable insights were found during the interviews regarding what Sony Mobile has gained from the Jenkins and Gerrit community involvement. During the analysis, the following innovation outcomes have been identified:

1. Free features.
2. Free maintenance.
3. Freed-up time.
4. Knowledge retention.
5. Flexibility in implementing new features and fixing bugs.

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 [40, 176]. For projects maintained by Sony Mobile, such as the Gerrit-trigger plug-in, a noticeable amount of external commits can be accounted for. Similarly, in communities where Sony Mobile is not a maintainer, they can still account for free work, but it 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. Furthermore, the use of tools (Jenkins and Gerrit) helped software developers and testers to better manage their work-flow. Consequently, it ***freed-up time*** for the developers and testers that could be used to spent on other innovation activities. The observed innovation example in this case was the developers working with OSS communities, acquiring and integrating the external knowledge into internal product development.

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 let us say we have 50 active developers, it's hard for the tech organization to compete with that kind of workforce and these developers at Gerrit are really smart guys. It is hard to compete for commercial Organizations"*. Further on, *"...We are mature enough to know that we lose 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 previous studies [130, 176]. Otherwise, the knowledge tied to I3 would have risked being lost for Sony Mobile.

Sony Mobile had many proprietary tools before opening up. 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 organization, 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. This was despite the required extra involvement and cost of more internal man-hours. 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 [131].

I1 stated that besides the flexibility, the Tools department is currently able to make a *"... more stable tools environment [at 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 it is 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 an 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 obtained external knowledge about the different parts of 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 organization 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 [117]. 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 [184]. The tool chain is even seen as the foundation for a platform which is supposed to facilitate this sharing [116]. 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"*.

Conclusion: Some of the innovation outcomes attached to Sony Mobile's openness entail more freed-up time for developers, better quality assurance, improved product releases and upgrades, inner source initiatives and faster time to market.

6 Results and discussion

Results from the quantitative and qualitative analysis are discussed below, of which the latter is addressed per theme, and connected to the research questions defined in Table 1. Table 8 presents the mapping of research questions to answers with section numbers. Furthermore, a brief summary of answers to research questions is highlighted in section 5.

Table 8: Mapping of answers to RQs with section numbers

Research questions	Answers to RQs
RQ1	Section 6.1
RQ2	Section 6.2, 6.7
RQ3	Section 6.3
RQ4	Section 6.6
RQ5	Section 6.4, 6.5

6.1 Involvement of Sony Mobile in OSS Communities

Addressing **RQ1** in Table 1, the quantitative analysis showed that Sony Mobile has an active role in numerous OSS projects. In most of the analysed projects, Sony Mobile is the initiator and maintainer. An exception is Gerrit where they entered an already established project. However, with 8.2 % (see Table 6) of the commits during the investigated time-span, they have established themselves in the community and been able to contribute the necessary adaptions for Gerrit to function as a part of the continuous integration tool-chain used inside Sony Mobile. This shows that Sony Mobile has an open mindset to creating their own OSS projects, as well as getting involved and contributing back in existing ones. In the projects which Sony Mobile has released themselves, they further show that they are open for contributions by others. In the Gerrit-trigger plug-in for example, they only represent 65% of the total commits. This also gives a clear picture of the help

gained by the external workforce as highlighted by OI. By opening up the Gerrit-trigger plugin and making it a part of the Jenkins community, they earn benefits such as shared feature development, maintenance and quality assurance. A reason why some of the other projects have fewer external commits (e.g., PyGerrit, Build-failure-analyzer and Team-views) may be that they are not as established and attractive for others outside Sony Mobile. A further explanation could be that Sony Mobile has not invested the time and attention needed in order to build successful communities around these projects.

6.2 Opening Up

In relation to **RQ2**, the move to Android took Sony Mobile from a closed context to an external arena for OI, recalls the description provided by Grotnes [73]. 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 [188] 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 the OI perspective, these activities can be seen as 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 [66]. 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, in contrast to the Android platform and its Open Handset Alliance, which is invite-only [73].

The quantitative results provide further support for the hypothesis that both established, larger corporations and small scale software organizations are involved in the development of Jenkins and Gerrit (see Table 6). Some of the small organizations are Garmin, Ostrovsky, Luksza, Codeaurora, Quelltextlich etc. This confirms findings from the existing OI literature, e.g. [77, 173] that other community players also can use these communities as external R&D resources and complimentary assets to internal R&D processes. One possible motivation for start-ups or small scale organizations to utilize external R&D is their lack of in-house R&D capabilities. Large scale software organizations exploit communities to influence not only the development direction, but also to gain a good reputation in the community as underlined by prior studies [42, 77].

Gaining a good reputation requires more than just being an active committer. Stam [173] separates between technical (e.g. commits) and social activities (e.g. organizing conferences and actively promoting the community), where the latter 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 commits. Along with the growth of commits, they have also matured in their commit strategy. As described in Section 5.2, the intent was originally to keep the Gerrit-trigger plug-in enclosed. This form of selective revealing [76] has however been minimized due to a more open mindset. As a consequence of the openness more plug-ins were initiated and the development time was 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. [194]. 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 [84]. The fear to reveal intellectual property was resolved thanks to the introduction of an OSS review board that involved both legal and technical aspects. Having an internal champion to give leverage to the needed organizational and process changes, convince skeptical managers [77], and evangelism of open source was a great success factor, also identified in the inner source literature [121].

6.3 Determinants of openness

When discussing if something should be made open or closed (**RQ3**) in Table 1, 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 already 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 [113].

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 [180]. Wnuk et al. [194] reason 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 potentially high maintenance costs.

Sony Mobile is aware that increased mobility [29] 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 [29]. 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 [115] Sony Mobile can use their contributions to steer and influence the direction of the community.

6.4 Requirements engineering

Tracing back to **RQ5** in Table 1, 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) [163]. This applies both internally (as a supplier and an administrator of the tools), and externally (as a member of the communities). From an RE perspective, they are their own stakeholder, competing with other stakeholders (members) in the Jenkins and Gerrit communities. These are important characteristics as stakeholders who are not developers are often neither identified nor considered [7]. A consequence otherwise could be that certain areas are forgotten or neglected which stands in contrast to Wnuk et al. [194] who state that adoption of OI makes identifying stakeholders' needs more manageable. Further, this brings an interesting contrast to traditional RE where non-technical stakeholders often need considerable help in expressing themselves. The RE in OI applied through OSS can be seen as quicker, light-weight and more technically oriented than traditional RE [163].

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 respect due to the Tools department involvement inside these communities [42]. Due to their high commitment and good track record, Sony Mobile employees have reached a high level in the governance organization. The Tools department combines these positions in the communities together with openness in terms of helping competitors and interacting in social activities [173] (e.g. developer conferences [102]). One reason for this is to attract quiet stakeholders, both in terms of influencing the community [40], but also to get access to others' knowledge which could be relevant for Sony Mobile. An example of this is the introduced

focus on scalability in both the Jenkins and Gerrit communities, where the Tools department needed to find stakeholders with similar issues to raise awareness and create traction to the topic. Communication in this requirements value chain [61] between the different stakeholders, as well as with grouping can be deemed very ad-hoc, similar to OSS RE in general [163]. This correlates to the power structure and how influence may move between different stakeholders.

Social interaction between the stakeholders is stressed by Panjer et al. [144] as an important aspect to resolve conflicts and to coordinate dependencies in distributed software development projects. The Tools department's 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 [164] is the preferable choice as it brings the core stakeholders together which allows for informal negotiations [61] and a live just-in-time requirements process [57], 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 of 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.

This use of live-meetings and social events for requirements communication and discussion, highlights the 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 for the individual stakeholder is to maintain or grow its influence and position in the governance ladder. Hence, organizations might need to revise their community involvement strategy and value what their intents are in contrast to if an online presence is enough.

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.

6.5 Testing

Addressing the **RQ5** in Table 1, we noticed during interviews that both Jenkins and Gerrit focus on manual test cases. At the same time, the communities started the transformation journey 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 plug-ins.

The Gerrit community is currently following the Jenkins' community patch, as stressed by interviewee 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. From this perspective, a trend may be seen 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 [76, 136].

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 the internal workload 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 [21]. 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 Table 5, the focus is on forward and re-engineering. An interesting concern is when and how much one should contribute 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 them. As discussed earlier, Sony Mobile has the strategy of focusing on issues which are self-beneficiary. Therefore, 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.

6.6 Innovation outcomes

In relation to **RQ4** in Table 1, the focal point of the OI theory is value creation and capture [31]. 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. [50] confirmed 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 in the literature. As expected, we found that Sony Mobile does not have established mechanisms in place to measure their performance before and after the Jenkins and Gerrit introduction. 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.

By taking an active part in the knowledge sharing and exchange process with communities [40, 176], 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. By extension, these software improvements may be labeled as product innovations depending on what definition to be used [50]. 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 [40]. The interviewees admit to that even a large scale software 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 Joy's law [107] *"No matter who you are, not all smart people work for you"*. Hence, it is vital to reach work force beyond organizational boundaries when innovating [31], and knowledge is still retained even if 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 [15] enables early identification of integration issues as well as increases the developers' productivity and release frequency [172]. With this reasoning, as reported elsewhere [117], 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 the possibility to tailor and be flexible that OSS development 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 [128]. Reduced time to market and increased quality of products are among the visible business outcomes. However, these outcomes cannot be confirmed due to a lack of objective metrics and came up as a result of interviews.

Another process innovation, which could also be classified as an organizational innovation outcome [4] is the inner source initiative. This initiative not only helps Sony Mobile to spread the tool chain, but also to build a platform (i.e. software forge [116]) for sharing built on the tool within the other business units of Sony Mobile. This may be seen as an intra-organizational level OI as described by Morgan et al. [130]. By integrating the knowledge from other domains, as well as opening up for development and commits, 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 [130]. Since Sony Mobile is a multinational corporation with a wide spread of internal culture, organizational changes are context and challenging.

6.7 Openness of tools software vs. Proprietary software

A specific aspect of **RQ2** in Table 1 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"*. This raises a discussion point of why Sony Mobile limits its openness to noncompetitive 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 study performed on the area [141].

In the light of the mapping study, it would be fair to state that the SE literature lacks studies on OI [141]. Organizations have a tendency to open proprietary products when they lose their value, and spinning off is a one way of re-capturing the value by creating a community around it [120]. This implication paves the 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 address the lack of contextualization weakness of OI mentioned by Mowery [133]. 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). In future work we plan to explore that situation to see if there are other patterns in these OI processes.

7 Conclusions

This study focuses on OI in SE at two levels: 1) innovation incorporated into Jenkins and Gerrit as software products, and 2) how these software improvements affect process and product innovation of Sony Mobile. 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 organization's product or service offering, e.g. as a platform, component or full product [180]. 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 committers 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,

Sony Mobile makes projects or features open source, which are neither seen as a main source of revenue nor as a competitive advantage **(RQ3)**.

In relation to the main innovation outcomes from OI participation (**RQ4**), we discovered that Sony Mobile lacks quantitative 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 commits from communities, regards the possibility 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 internal resources 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 the feature in question and individual needs of the respective parties. Further, prioritization is made in regards to how an issue or feature may be seen as beneficial, in contrast to 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 organizational standards.

The scope of the study findings is limited to software organizations with similar context, domain and size as Sony Mobile. It is also worth mentioning that the involvement of stakeholders in the Jenkins and Gerrit OSS communities suggests that the continuous integration processes of these OSS projects are comparable to the corresponding process at Sony Mobile. Thus, we believe that findings of this study may also be applicable to incumbents as well as small software organizations identified in this study.

Future work includes investigation of other contexts and cases where companies use OSS aiming to leverage OI, and to cross-analyze the presented findings in this paper with findings from future case studies.

APPENDIX C

SUPPLEMENTARY INTERVIEW QUESTIONNAIRE

Demographics

- Where do you work?
- What is your job title?
- Which department do you work for in the organization?
- How many years of experience do you have?
- Could you, in short, describe your daily work and responsibilities?

General involvement

- Are you, or have been, in any way actively involved in any open source community in your daily work? (Gerrit, Jenkins, any other?)
- Could you describe your involvement?
- What is/was the reasons for your involvement in these open source communities? (Volunteered or tasked by management?)
- How much time are you allowed to spend on community interaction?
- How is your involvement with these community in your spare time, outside of your daily work?
- What development process/methodology do you use and how does it interact with the community? (process of working)

Requirements

- What are the sources (internal and external) behind the requirements/features? (by tool developers, tool users, pm's, othersâĂę)
- How do you manage and implement the requirements/features?
- How are the requirements approved and prioritized? (By developers alone, pm's, communityâĂę)
- How is your involvement perceived from the community? Positive or negative? How come? (competitors)
- Are there any internal (organizational) obstacles in contributing to the community? (Time, IP, managementâĂę.)
- Are there any external obstacles related to the involvement in the community related to the addition of new requirements/features?
- How did you overcome these?

Testing

- How does your internal process of reporting bugs differ from the community's? (tools for reporting bugs in community)
- How do you manage traceability between tests and requirements?
- Who is responsible for fixing those bugs? (Process behind, consequence on quality and resolution time)
- How does your internal process for correcting bugs or issues, differ from the community's?
- Are there any obstacles related to the involvement in the community related to the testing process? How did you overcome these? (Communication, synchronized level of quality/tests between contributors)

Business/strategy

- What motivates your organization to contribute to open source project(s)? (Beyond lower cost, improved quality?)
- What is the strategy behind these commits?
- Did you consider alternate strategies such as buying proprietary tools (COTS) or hiring people/outsourcing for the development these tools? Why?
- How are these strategies supported by your internal procedures (IP department)?
- Is it a local strategy or global strategy? Who are the sponsors?
- How has the commits effected the relation with other (corporate) stakeholders in the communities? (Free-riding, governance structure, constraints, Sony Authority, collaboration, balance between community and Sony's needs, community buildup)
- How has the commits effected the relation with other competitors? (Free-riding, governance structure, collaboration)

Perception on innovation and outcome

- How has the usage/development of these tools effected the Sony Mobile's product development? (Developers, testers)
- How has the usage of these tools effected the products?
- Is innovativeness of a requirement/issue/bug considered, and if so, what effect does it have on the requirements and contribution process?
- How has the involvement in the communities implicated on innovation in your: 1) Processes? 2) Products 3) Organization 4) Business strategies
- How do you measure the impact from the development/usage of these tools on Sony Mobile's product development? Metrics etc.
- Is the knowledge gained from the OSS tool development transferred and exploited outside of the tools development? (Absorptive capacity âĂŞ Firm level, individual level)

Ending remarks

CHAPTER III

A THEORY OF OPENNESS FOR SOFTWARE ENGINEERING TOOLS IN SOFTWARE ORGANIZATIONS

Abstract

Context. The increased use of Open Source Software (OSS) affects how software-intensive product development organizations (SIPDO) innovate and compete, moving them towards Open Innovation (OI). Specifically, software engineering tools have the potential for OI, but require better understanding regarding what to develop internally and what to acquire from outside the organization, and how to cooperate with potential competitors.
Aim. This paper aims at synthesizing a theory of openness for software engineering tools in SIPDOs, that can be utilized by managers in defining more efficient strategies towards OSS communities.
Method. We synthesize empirical evidence from a systematic mapping study, a case study, and a survey, using a narrative method. The synthesis method entails four steps: (1) Developing a preliminary synthesis, (2) Exploring the relationship between studies, (3) Assessing the validity of the synthesis, and (4) Theory formation.
Result. We present a theory of openness for OSS tools in software engineering in relation to four constructs: (1) Strategy, (2) Triggers, (3) Outcomes, and (4) Level of openness.
Conclusion. The theory reasons that openness provides opportunities to reduce the development cost and development time. Furthermore, OI positively impacts on the process and product innovation, but it requires investment by organizations in OSS communities. By betting on openness, organizations may be able to significantly increase their competitiveness.

Table 1: Definition of terms used in the study

Term(s)	Definition(s)
Pecuniary	Pecuniary resources refer to competitive assets, which are revealed against financial rewards, for example licensing proprietary software tools [30, 39].
Non-pecuniary	Non-pecuniary resources refer to non-competitive assets, which are revealed without immediate financial rewards for a company [30, 39], for example, OSS tools (e.g., Jenkins and Gerrit) used in the development of proprietary products.
Outside-In (or Inbound) innovation	The inward flow of knowledge from outside the boundaries of an organization to inside [39, 66, 189], for example, the extraction of knowledge from OSS communities.
Inside-out (or Outbound) innovation	The outward flow of knowledge from inside the boundaries of an organization to outside [39, 66, 189], for example, sharing the internal knowledge of an organization to OSS communities.
Coupled Innovation	The outward and inward flow of knowledge from the boundaries of an organization [39,66,189], for example, acquiring and sharing the knowledge to OSS communities.

1 Introduction

The introduction of Open Source Software (OSS) in commercial settings have opened new possibilities for innovation in software-intensive product development organizations (SIPDOs)[1]. This shift implies that the internal research and development (R&D) is no longer the only strategic asset for the companies in creating products and services. Access to, and interplay with, external sources and actors provide new opportunities but also create new challenges.

One specific type of OSS is software engineering tools used in the development of software-intensive products. The tools themselves are not the core business of the SIPDOs, but they rely heavily on them to be efficient in their software development. Further, the costs of improving the tools and keeping them up to date may be significant, and thus SIPDOs may want to share the costs with other organizations.

[1] SIPDO refers to organizations developing products or services with a substantial amount of software defining the product/service behavior, mostly embedded in physical products.

Table 2: Huizingh's classification of openness [83]

Innovation Process	**Innovation Outcome**	
	Closed	**Open**
Closed	Closed Innovation	Public Innovation
Open	Private Open Innovation	Open Source Innovation

In 2003, Chesbrough proposed the term Open Innovation (OI), later defined as *"a distributed innovation process based on purposively managed knowledge flows across organizational boundaries, using pecuniary and non-pecuniary mechanisms in line with the organization's business model"* [30]. Cheshbrough's definition of openness hints at valuable ideas that can emerge and commercialize from inside and outside the organization. OI entails various activities, e.g., inbound (also called inside-out), outbound (also called outside-in) and coupled activities [66], and each of these activities can be more or less open. Dahlander and Gann [39] defined inbound versus outbound OI, and pecuniary versus non-pecuniary interactions. Researchers have used the term inside-out/outbound and outside-in/inbound synonymously in the OI literature. The terms used in this paper are defined in Table 1.

This paper uses the openness classification by Huizingh [83] who categorized processes and outcomes as closed or open, see Table 2. Open processes deal with either using the input from outside the organizations, or by externally exploiting an internally developed innovation. This is in contrast to closed processes, where the innovation process is kept in-house [83]. On the other hand, open outcomes entail devoting the scarce resources to innovation, and then giving away the outcome (e.g., proprietary solutions) for free to OSS communities, in contrast to closed outcomes where organizations keep their solution in-house.

OI has two important implications on the organizational structures [39]: 1) globalization enables increased division of labor, and 2) introduction of OSS enables professionals to seek for portfolio careers instead of working for a single employer. Therefore, SIPDOs need to find alternate ways of finding talented workers that might not be interested in exclusive or direct employments. Furthermore, intellectual property rights (IPR) utilization, technology standards, and venture capital allow software organizations to trade ideas.

Bosch [23] claims that speed, data and ecosystems are the main factors that impact SIPDOs in their software engineering practices. At the same time, the size of software-intensive products continues to grow. This growth incurs the need for faster and better adoption of applications, technologies, components, services, and ecosystem partners. In order to address this challenge, SIPDOs may utilize OSS communities to find more resources to increase the speed and reduce the development and maintenance costs. Linåker et al. [118] presented a Contribution

Acceptance Process (CAP) model, which provides *operational* guidelines for SIPDOs regarding when and what features to contribute to OSS communities. CAP underlines four objectives for OI adaption: cost saving, time-to-market, control in the ecosystem, and strategic alliances and investments.

However, we have found no guidelines for SIPDOs in order to make strategic decisions regarding *OSS tools*, i.e. what role in Huizing's taxonomy to choose in the open innovation (i.e. open processes, open outcomes), for OSS tools which are not the core business (non-pecuniary) for the organization (e.g., OSS tools like Jenkins and Gerrit) but are vital to support the internal product development.

We present a *theory of openness for software engineering tools* in software organizations that complements and expands our previous research efforts [137, 141] and provides the necessary organizational aspects that support SIPDOs in their transformation towards OI. The scope of this study covers the use of non-pecuniary OSS tools in organizations' proprietary software development for outside-in and inside-out innovation (i.e. coupled innovation). This study focuses on the strategic role of OSS tools in an organization, where we use software build tools as cases, due to their strategic role in the build chain [137] [App D].

We synthesize the theory from two previous empirical studies [137, 141], complemented by a survey in the Git, Gerrit and Jenkins communities [App D]. The theory provides *strategic* guidelines and helps SIPDOs to adopt OI tools in relation to reduced development cost, shorter time-to-market and process, and product innovation.

The paper is structured as follows. Section 2 presents related work for this study, section 3 provides an overview of the research methodology, and section 4 shows the research synthesis process. Section 4 describes the narrative synthesis process, followed by section 5, which explains the theory of openness for tools in software engineering. We conclude the paper in section 5.

2 Background studies and related work

This work is conducted in the context of OI and OSS, which is explored in section 2.1. We define theory for software engineering, and summarize thus related work on theory in SE in section 2.2. Our recent systematic mapping study [141] and the case study on OI using OSS tools [137], as well as the validating survey presented in Appendix D are summarized in section 2.3.

2.1 Open innovation using OSS

SIPDOs get involved in OSS communities to leverage their internal development resources, to steer the communities towards organizations' interest, to reduce time-to-market and development costs [137, 141]. Increased openness towards OSS communities constitutes a challenge of keeping companies' competitive advantage while contributing software to OSS communities.

Our systematic mapping study [141] identified some of the OSS contribution strategies, such as engaging in OSS communities [41], open business models [27], and adopting selective revealing [76]. Furthermore, the knowledge acquired from OSS communities helps organizations to improve internal development processes, e.g., adopting continuous integration, leading to faster releases etc, which leads to product innovation [117].

However, it should be pointed out that OSS is not OI by definition. In order for an OSS project to be a representative example of OI, it is necessary that OSS is incorporated into the organization's business model and contributes to value creation in the organization's product development [137]. Thereby, OSS can be used to implement an OI strategy.

2.2 Theory building in SE

Theories offer common conceptual frameworks for more precise and structured organization of facts and knowledge. Theories also facilitate communication of ideas and knowledge through analytical generalization [165, 198]. Many authors in the SE research community have raised concerns about lack of theories in SE [14, 20, 53, 80, 162, 169, 177] and pointed on the limited nature of the work about SE theories.

The theory of distances by Bjarnason et al. [20] is an example creating a theory in SE [175]. Like their study, we also follow the guidelines provided by Sjøberg [169] defining four theory constituents: the constructs (basic elements), propositions (interaction of constructs), explanation (reasons for propositions specification) and scope (application of the theory).

2.3 Background studies

Three studies constitute the background for the theory creation: a systematic mapping study (S1), a case study (S2), and a survey study (S3).

S1 – A systematic mapping study was conducted to investigate OI in software engineering [141]. The findings suggest that a large majority of the studies have industry relevance. Therefore, we conclude that OI in software engineering is oriented towards industry practice. At the same time, the majority of studies were evaluation studies. This indicates a research gap in terms of conducting more solution and validation research.

Furthermore, the mapping study indicates that managers have too little time and resources to invest in significant contributions to OSS communities, even when an organization has a well defined OSS contribution strategy. Another important hurdle for organizations in embracing openness is the risk of losing intellectual property rights to competitors. In order to deal with it, business strategies are important to reveal a competitive asset and to avoid committing extra resources to

commodity parts of the software. Thus, organizations can pursue a differentiation strategy with a controlled degree of openness.

The key message of the systematic mapping study is that the OI does not replace the existing R&D capabilities of an organization. Rather, it complements or boosts the existing R&D activities of an organizations to accelerate the innovation process.

S2 – The Gerrit/Jenkins case study was conducted to explore Sony Mobile's participation in the Jenkins and Gerrit communities and to highlight the innovation outcomes attached to coupled (inside-out and outside-in) OI processes [137]. The study describes the shift of Sony Mobile from using a closed innovation model to an open innovation model. This shift allowed Sony Mobile to utilize the Jenkins and Gerrit communities and improve their internal software development processes. We discovered that both small and large scale organizations are involved in the Jenkins and Gerrit communities along with Sony Mobile. However, Sony Mobile is the largest contributor in the development of aforementioned tools.

The key trigger that embraced openness at Sony Mobile was the paradigm shift from Windows to Linux development environments. However, this openness was limited to the tools that are not considered a direct source of revenue. Furthermore, they used dedicated internal resources (within the tools department), to gain influence and steer OSS communities towards its own business interests.

The study also underlines the lack of quantitative metrics to measure the effect of OI adaption on Sony Mobile's innovation capacity, but the qualitative data suggests more stability and flexibility in the internal development environment.

S3 – The survey in Jenkins, Gerrit and Git communities was designed to validate the findings from the case study, especially with respect to generalization to other SIPDOs. A questionnaire was distributed in the Jenkins, Gerrit and Git communities. Respondents include employees working in SIPDOs and OSS communities simultaneously and were classified as either users or contributors to OSS communities.

The results show that SIPDOs participate in OSS communities to bring in innovative ideas, save development costs, get the latest patches, gain control and influence communities for the organization's own process and product innovation. However, 62 % of the respondents spent less than ten hours per week to work with OSS communities, which indicates that software organizations may not have fully understood the benefits of openness yet. The details from the survey are presented in Appendix D.

3 Research design

Most research in software engineering related to OI focuses on exploration of the OI notion **[S1]**. We identified a need to synthesize the existing knowledge in order to propose a solution to the problems faced by software organizations due to the

Table 3: Studies used as a theoretical frame of reference

Study_ID	Study_name	References
S1	Open innovation in Software Engineering: A Systematic Mapping Study [141]	[24,40,42,46, 49,76,84,106, 130, 131, 145, 156, 168, 173, 176, 182, 187, 194]
S2	Open Innovation using Open Source Tools: A Case Study at Sony Mobile	[137]
S3	Survey on OI	[App D]

adoption of OI using OSS tools. This study aims to find common themes among the three background studies [**S1**, **S2** and **S3**], using a synthesis method adapted from Popay et al. [152] and Cruzes et al. [38].

3.1 Research goals

The scope of the research encompasses the use and impact of OSS tools on the internal software development of SIPDOs. First, we focus on the synthesis of factors associated with OI for OSS tools in SE literature. Second, we derived the theoretical constructs for OSS tools in SIPDOs from the synthesis. The first research goal is addressed in Sections 4.2 to 4.2, while the second research goal is addressed in Sections 4.2 and 5.

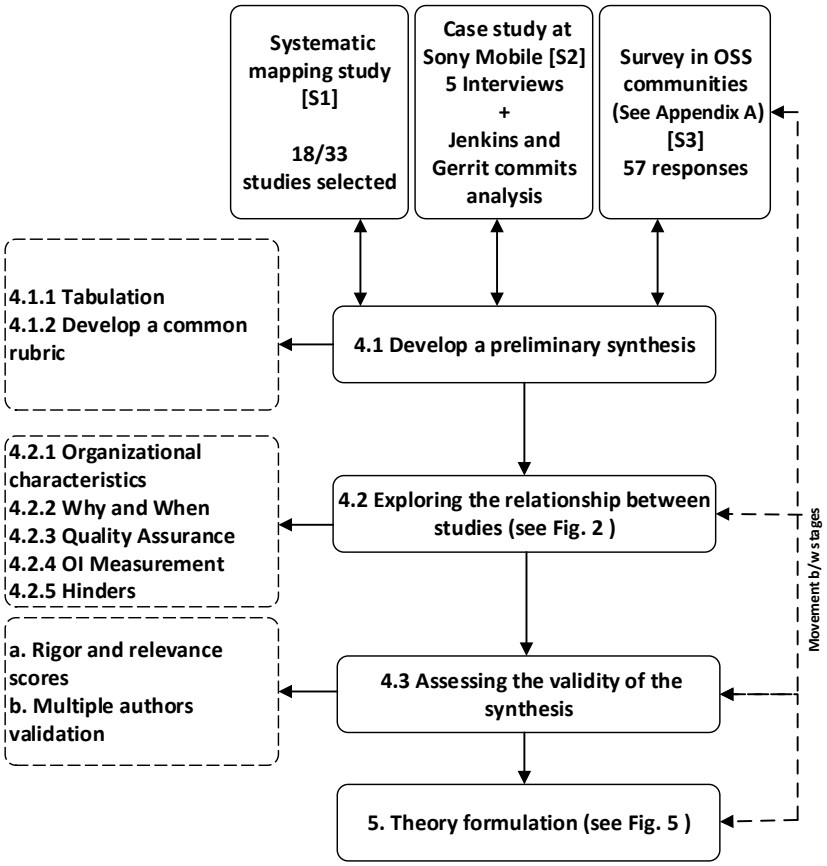
3.2 Narrative synthesis design

We performed narrative synthesis [37, 152], as summarized in Figure 1. The synthesis integrates the systematic mapping study [S1], the Sony Mobile case study [S2], and the Git, Jenkins and Gerrit survey [S3], see Table 3. Furthermore, the findings of **S1** were taken into account to design **S2** and **S3**. The details of each step in Figure 1 are explained in Section 4.

It is to be noted that **S1** is a mapping study which contains 33 primary studies, addressing open innovation in SE. However, only 18 studies were selected from **S1** for this work, which pertains to the scope of this study, and thereafter included in the synthesis process. The selected studies are listed in Table 3 next to **S1**. The selection criteria of the studies from **S1** are as follows.

- Studies pertaining to the scope of OSS communities used by SIPDOs.

Figure 1: Synthesis method and theory formulation



- Studies addressing the integration of external knowledge with internal knowledge (outside-in) in organization's software product development.
- Studies highlighting selective revealing (inside-out) of the internal knowledge to OSS communities by SIPDOs.
- Studies discussing the investment of organization's internal resources in OSS communities.
- Studies reporting the factors that encourage SIPDOs to choose openness in terms of processes and outcomes in relation to Huizing's classification (Table 2).

4 Narrative synthesis

The empirical evidence on OI for SE tools is synthesized according to the research design steps presented in Figure 1.

4.1 Developing a preliminary synthesis

We developed preliminary synthesis based on the identified data. We performed an initial data assessment and concluded that the data is predominantly qualitative [152]. The pre-synthesis qualitative data comes from the systematic mapping study [S1] and the case study [S2]. Furthermore, a mix of quantitative and qualitative data is extracted from the survey study [S3]. The survey was distributed using Google forms and thus responses were stored in a Google sheet, which was downloaded to analyze the responses

Tabulating the data

The first author extracted the data related to the first research goal, presented in Section 3.1. The data was stored from **S1**, **S2** and **S3** into a separate spreadsheet. The preliminary data extraction criteria include the organizations, the rationale behind adopting OI in software organizations, strategies, and the challenges faced by the organizations while working with OI. In 1, Table 3 shows a part of raw data collected from **S1**, **S2** and **S3**.

Outcome: The outcome of this step is a table of raw data extracted from **S1**, **S2** and **S3**.

Developing a common rubric

The first author defined a common rubric for all three studies. Next, the first author repeated the data extraction process from **S1**, **S2** and **S3** to ensure correctness and look for additional evidence. Initially, we did not find all the themes mentioned in

Table 4: Definition of the joint rubrics for the three studies (S1–S3)

Themes	Definition
1. Who	What characterizes the organizations involved in open innovation using OSS tools e.g., Jenkins, Gerrit, Git?
2. Why	Key factors considered by software organization before adopting openness
3. When	What are the strategies used by software organizations to be open ?
4. Quality assurance	What are the tools used to automated test execution and the key challenges attached to software quality assurance in open innovation?
5. OI measurement	What metrics are used by software organization to measure the impact of Open innovation on development process?
6. Hinders	Challenges faced by organizations to adopt OI

Table 4. For example, the quality assurance and and OI measurement themes were not obvious, but after collecting the data, these two new recurring patterns were identified.

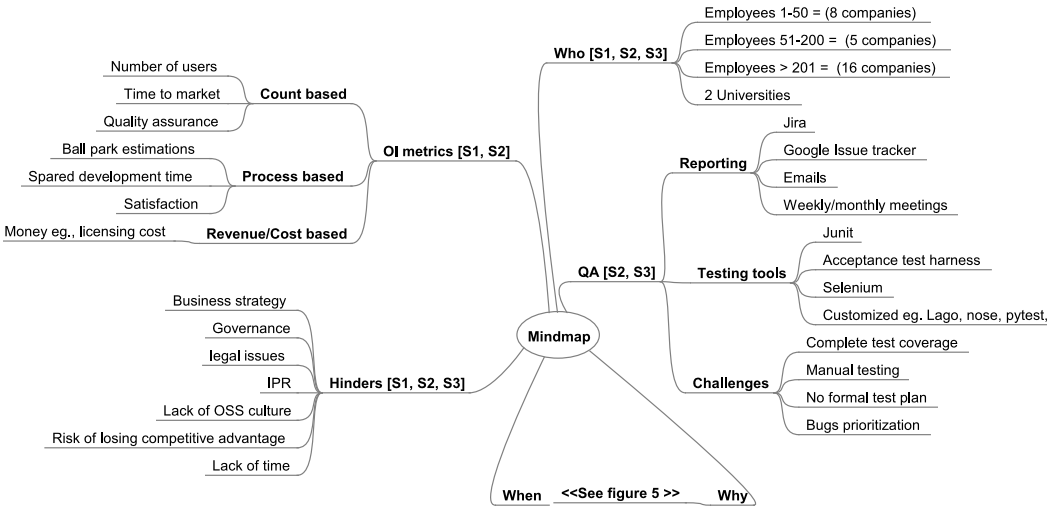
Outcome: The result of this step leads to Table 4, which highlights the six joint rubrics from the preliminary data extraction process.

4.2 Exploring the relationships between the studies

We used the common rubrics outlined in Table 4 to identify the common characteristics between the systematic mapping study [S1], the case study [S2] and the survey in OSS communities [S3]. We wrote a short summary for each of the rubrics (tabular textual description) of each study [S1,S2,S3]. This allows accuracy and consistency checks with previous steps and helps to draw aspects from individual studies that may not have seemed relevant at the start of synthesis, but have become of interest during the subsequent stages of the synthesis.

Outcome: This step leads to the mind map presented in Figure 2. Each of the categories mentioned in the mind map is discussed below.

Figure 2: Mind map of relationship between studies



Organizational characteristics (Who)

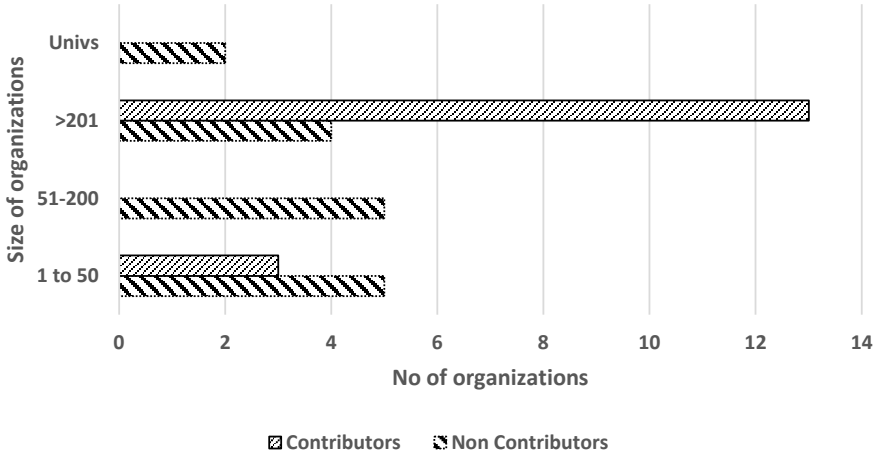
Both small and large companies use OSS tools as a means for OI. Figure 3 shows the distribution of companies in **S2** and **S3** contributing or using OSS tools in their internal product development. The contextual details of the organizations such as name, product type, size and OSS tools can be found in F. From the contextual details of the studies in **S1**, it is not obvious whether they are contributors or non contributors. Therefore, the companies included in this study are extracted from **S2** and **S3** only.

Figure 3 shows that 16 software organizations are not only involved in using but also contributing back to OSS communities, such as Jenkins, Gerrit and Git. Out of the 16 contributing software organizations, 13 have more than 200 employees, see Figure 2. This confirms that not only small software organizations are using OI but also larger organizations realize the importance of contributing to OSS communities to improve their internal development process. This observation is in line with the concept of process innovation [117] and the coupled open innovation process (inside-out and outside-in [54].

Why – When

We explore the relation between the two themes entitled *Why* and *When* by creating a 2x2 matrix depicted in Figure 4, which is referred as a visual representation of *Why* and *When*.

Figure 3: The number of contributors vs non contributors in S2 and S3, distributed over size categories



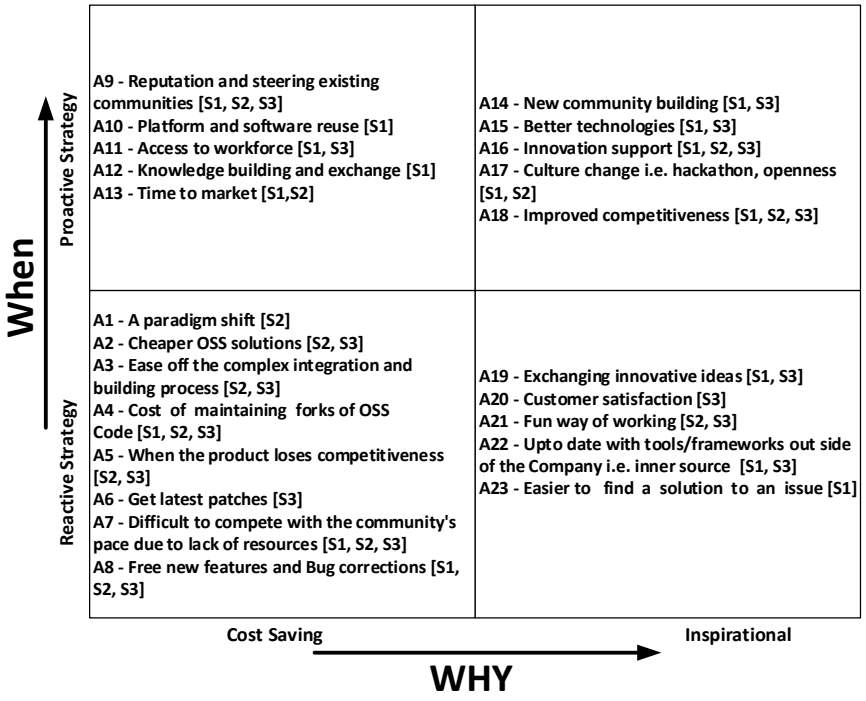
The horizontal axis of the model shows the two main driving forces related to the motivations for adapting OI; 1) Cost saving and 2) Inspiration. Firstly, the **cost saving** driving force refers to the aim of reducing the product development cost. Examples of cost saving entail incorporating an OSS solution instead of making it in-house, or spending more resources on differentiating features instead of on commodity features. The **inspirational** driving force refers to software engineers taking initiatives on their own to optimize the daily software engineering workflow by embracing openness. A typical example of inspiration is the employees working for SIPDOs that develop proprietary solutions, but also actively contribute to seek better solutions to OSS communities using the outside-in OI principle [54].

The vertical axis in Figure 4, distinguishes between reactive and proactive OI adoption strategies. The **reactive** OI adoption strategy considers reacting to events rather than taking the leading role. In other words, the reactive strategy prevents software organizations from taking initiatives and they mostly adopt the wait-and-see strategy. On the contrary, the **proactive** OI adoption strategy enables organizations to anticipate what the future will be, and to react accordingly before a threat actually happens.

Empirical evidence of the aforementioned strategies is combined with the cost factor in Figure 4. Each attribute in Figure 4 can be traced back to its original study **S1**, **S2** or **S3**, and each quadrant is described in detail below.

Reactive strategy – Cost saving. The reactive strategy in relation to cost saving entails cost reduction of the development activities. The following are the

Figure 4: Visual representation of Why and When



attributes along with the definitions extracted from **S1**, **S2** and **S3**.

A1 – A paradigm shift [S2]: Refers to the switch from Windows based software development environment to Linux [137].

A2 – Cheaper OSS solutions [S2, S3]: It is more cost effective for software organizations to adopt OSS code (even when integration efforts are substantial) instead of developing an in-house solution from scratch [137], [App D].

A3 – Ease off the complex integration and building processes [S2, S3]: The introduction of OSS tools (Jenkins, Gerrit etc.) made the continuous integration process easier for software developers and testers [137], [App D].

A4 – Cost of maintaining forks of the OSS code [S1, S2, S3]: To fork off the OSS code and its internal maintenance only makes sense if it adds significant value. In case of commodity software, it does not give any business advantage. Therefore, contributing commodity parts of the products alleviates software organizations from patching the code [137, 156], [App D].

A5 – When the product loses competitiveness [S2, S3]: Refers to making the project OSS in order to attract interest from the community and receive contributions as well. Making a product that loses competitiveness OSS also opens up for

alternative revenue sources. Keeping the software closed only causes additional maintenance costs with no business benefits [137], [App D].

A6 – Get the latest patches [S3]: This is also connected to costs of maintaining forks of OSS code. Organizations make the commodity code open to share the patching cost with the OSS community instead of spending unnecessary resources to maintain it internally [App D].

A7 – Difficult to compete with the community's pace due to lack of resources [S1, S2, S3]: OSS communities are often comprised of skilled development work forces from around the world. It is difficult for software organizations to find such resources and hire them all [40, 137], [App D].

A8 – Free new features and bug corrections [S1, S2, S3]: Refers to software organizations actively participating in OSS communities and in return receiving new features and free bug corrections to facilitate the development process [46, 76, 137, 182, 194], [App D].

Summary. A factor that leads organizations to adopt the *reactive cost saving strategy*, is the substantial costs of proprietary tools vs. the much lower cost of using OSS tools. Therefore, many software organizations choose to switch from Windows to Linux development environments to ease off the complex source code integration and building processes [137]. Factors that motivate organizations to adopt reactive cost saving strategies include patching cost, products losing their competitive advantages, to get new features, licenses that demand organization to contribute back, and difficulty to keep up with ever-growing OSS communities [Appendix D]. Forking an OSS solution leads to internal maintenance of commodity software. As the core of the reactive strategy is to save maintenance costs, organizations choose to open up commodity solutions and share the maintenance cost with all stakeholders in the community.

Proactive strategy – Cost saving. The following attributes are extracted in relation to the proactive cost saving strategy.

A9 – Reputation and steering existing communities [S1, S2, S3]: Refers to becoming an active contributor in OSS communities and influence or steer these communities towards organizational interests [42, 137, 176], [App D].

A10 – Platform and software reuse [S1]: Refers to reusable software components used together with proprietary software in the products [130].

A11 – Access to workforce [S1, S3]: Refers to the utilization of smart development workforce, which does not work directly for an organization, but possible to utilize through OSS communities [42, 76] [App D].

A12 – Knowledge building and exchange [S1]: Refers to in-flows and out-flows of knowledge in software organizations [24, 40, 84, 106, 130, 173, 176, 187].

A13 – Time to market [S1, S2, S3]: Adopting the commodity code from OSS communities is not only cheaper but also reduces the time it would require to develop that code in-house [130, 137, 176] [App D].

Summary. The *proactive cost saving strategy* allows SIPDOs to engage in OSS communities and become their trustworthy member to steer communities toward their own business models and use the community developers for organizational focused development. This, in turn, makes it possible for organizations to access all the software developers, that exist beyond the organizational borders, without hiring them. Organizations are required to invest in existing communities to reduce the time-to-market and development costs by utilizing developers from OSS communities.

Proactive strategy – Inspirational. The following attributes are extracted in relation to the proactive inspirational strategy.

A14 - New community building [S1, S3]: This is one of the ways to further enhance the organization's internal innovative capacity. It is desired by organizations when existing communities are not fulfilling the expectations of the organizational needs [40, 42, 84, 131] [App D].

A15 - Better technologies [S1, S3]: Refers to building new features with the help of, or suggestions from, hundreds of developers in the community in relation to a handfull of developers inside the organization [130, 131, 173, 176] [App D].

A16 - Innovation support [S1, S2, S3]: Refers to complementing the organization's internal R&D process by indulging organizational resources into OSS communities and using this knowledge to improve the organizations' innovative capacity [84, 130, 137, 145, 176] [App D].

A17 - Culture change [S1, S2]: Deals with promoting the embracing of an openness culture in product and process development, without the fear of losing competitive advantage and being able to solve problems using hackathons [137, 168].

A18 - Improved competitiveness [S1, S2, S3]: This is primarily in relation to decreased time-to-market. Using OSS code helps organizations to get their products out in the market faster and thus increase competitiveness [130, 137, 176], [App D].

Summary. The *proactive inspirational strategy* driven by the managers provide inspiration to create new communication channels between developers belonging to different organizations. The objective is to build new OSS communities if the existing communities are not supporting the organization's internal innovation processes. Software development teams empowered by proactive strategies, search for innovative solutions and embrace an openness culture (e.g., hackathons) for the shared knowledge building. Consequently, the exploration of the proactive strategy frees up time for development teams to focus on differentiating tasks, which improves competitiveness and supports internal innovation.

Reactive strategy – Inspirational. The following are the definitions of factors relevant to inspire developers in an organizations by working with OSS communities.

A19 – Exchanging innovative ideas [S1, S3]: Refers to utilizing OSS communities as a forum for exchanging ideas and helping each other [40, 49, 130] [App D].

A20 – Improved customer satisfaction [S3]: Continuous development and software updates lead to improved customer satisfaction [App D].

A21 – Fun way of working [S2, S3]: Software developers involved in OSS communities find it a fun way of working. Furthermore, organizations also use communities to keep their workforce motivated by finding news way of working [137], [App D].

A22 – Up-to-date with tools/frameworks [S1, S3]: Allows software developers to stay up-to-date with the new tools/frameworks outside the organizational border, using OSS communities. In addition, it allows developers to acquire knowledge from OSS communities and integrate that across different units of an organization [130],[App D].

A23 – Easier to find a solution to an issue [S1]: Gives software developers in organizations the possibility to look for solutions to their problems beyond organizational borders [130].

Summary. The *reactive inspirational strategy* leads organizations to use OSS communities for the exchange of ideas and helps developers to find better solutions. To keep the motivation and to be up-to-date with the tools, organizations encourage their developers to engage in OSS communities. Furthermore, developers find it easier to find a solution to their problems by being open.

Quality Assurance

Quality assurance in relation to OI is an under-researched area, while there are several challenges in testing the OSS tools. For organizations who have to test OSS tools together with OSS communities (e.g., Jenkins, Gerrit), they identify the need of automated testing. However, it is still not possible to have a complete test coverage due to many configuration settings and the open nature of these tools, but automation helps developers to identify the defects quickly [139].

Out of the survey respondents [App D], 34.5 % highlighted that manual testing consumes too much time, and there is a need for an automated testing framework. The Jenkins community introduced an acceptance test harness [139], which is an automated test suite to test Jenkins core and its plug-ins. However, the Gerrit community is not as mature as the Jenkins community, and the community is trying to replicate the automated testing harness from Jenkins. Further details of quality assurance from the survey can be found in 2.

OI Measurement

We identified evidence regarding the metrics used by software organizations to measure the impact of OI as reported in Table 5.

Table 5: Metrics to measure Open Innovation

OI Measurement	Metrics	Definition
Count measures	Number of users [S3]	Number of users using OSS tools
	Time to market [S2, S3]	Measure the time difference with and without OSS tools
	Quality assurance [S3]	Frequency of bugs using with or without using OSS tools
Process measures	Ball park estimations [S2]	How much development uptime lost if the organization is not able to quickly fix the show stopper bugs
	Spared development time [S2,S3]	Time required for own developed solution vs OSS
	Satisfaction [S3]	Five point Likert scale from Not at all satisfied to somewhat satisfied
Revenue/Cost based measures	Money [S2, S3]	Licensing cost of OSS vs non-OSS tools

We categorized these measures based on the criteria provided by Edison *et al.* [50] and divided them into the three categories: 1) count measures (number of users, time taken to introduce new products in the market, etc.), 2) process measures (assess the innovation capability of an organization), 3) revenue and cost measures (money, licensing cost). It must be mentioned that these metrics came up as suggestions from interviewees [S2] and survey respondents [S3].

Hinders

Embracing openness is also associated with barriers. The lack of time, resources and understanding of OSS communities hinders organizations to commit the resources dedicated for OSS communities [S1, S3]. Although, software organizations recognize the benefits of participating in or creating a strong community, there are a few limitations associated with it. Those limitations entails management constraints due to the lack of funding, understanding, contributor agreements lawyers/significant internal paperwork, unfamiliarity, distrust with regards to OI contribution strategies of working [137][App D].

Table 6: Rigor and relevance scores for the studies

Ref.	C	D	V	Rigor Sum	U	S	RM	C	Rel. SUM
S2 [137]	1	1	1	**3**	1	1	1	1	**4**
S3 [App D]	1	1	1	**3**	1	0.5	1	1	**3.5**

Executives and R&D legal experts need to evaluate OSS contributions for licensing, protecting organization's IP rights and patent infringements [S2]. The value of being in control of an OSS community is not well understood [S3]. Moreover, a slow approval process [S2] for contributions also leads to lack of motivation among employees to contribute to OSS communities. This is also connected to working with the development methodology (Scrum) since there is no room in the sprints to prioritize OSS contributions [36, 137, 194].

4.3 Assessing the validity of the synthesis

We used the rigor and relevance criteria followed by multiple authors validation in which all authors were involved in validating the synthesis process to test the validity of the synthesis.

We applied the criteria, previously used by Munir et al [138] to the case study [S2] and survey [S3] underpinning this work, see Table 6. The rigor and relevance criteria used for [S2] and [S3] are available in 2.1 and 2.2. Scores between (0–1.5) are considered as low rigor, while high rigor entail scores of 2 or above. On the other hand, studies with the score from (0–2) have low relevance, while scores from 2.5 or above are seen as high relevance. The systematic mapping study [S1] was not scored based on the criteria because the criteria are not meant for literature reviews. However, the study **S1** itself used the rigor and relevance criteria to rate all the primary study used in the synthesis process.

Below, the four types of validity threats [155, 159] related to the synthesis are addressed with their mitigation strategies.

Internal Validity

Internal validity refers to causal relationships and the introduction of potential confounding factors [159].

Peer examination. The objective behind conducting the synthesis study was to generate theoretical guidelines based on the three studies that explored OI in software engineering. Since the data synthesis process entails qualitative data, this introduces the risk of subjectivity. In order to minimize this risk, the second and third authors were involved in validating the synthesis.

The theoretical foundation for the theory of openness for software engineering tools in software organization included studies **S1**, **S2** and **S3**. However, **S1** comprised of studies where the scope of the few studies was not limited to OSS tools only, while the theory of Openness is limited to OSS non-competitive tools. The underlying phenomena may be different and thus present a validity threat to the theory.

External Validity

External validity deals with the extent to which it is possible to generalize the study findings to other contexts [159]. Merriam [125] viewed external validity from the perspective of assumptions underlying qualitative research and proposed several reformulations of generalizations such as working hypothesis, concrete universals and readers or user generalizations.

In this study *concrete universals* [56] seems more applicable where a particular context is applied to similar contexts subsequently encountered. The context in this study refers to SIPDOs using OSS tools in their internal software development. To be specific, **S2** presents a case study at Sony Mobile using OSS tools (e.g., Jenkins, Gerrit, Git) in their proprietary software development and **S3** contains 26 organizations either using or contributing to OSS communities. The contextual details of each organization are presented in F. The broad representation of organizations that support the findings of this study, indicate generalizability to organizations using OSS tools for their internal software product development.

Construct Validity

This deals with choosing the suitable measures for the concepts under study. This study used two primary and 18 secondary studies from **S1** for narrative synthesis. Although, it is possible to use secondary studies for narrative synthesis, the difficulty in achieving the abstraction of results at the same level presents a validity threat. In order to minimize this risk, studies **S2** and **S3** were conducted by taking inputs from the findings of **S1**, which enables researchers to find common relationships among studies for the comparison (see Table 4).

The survey study is susceptible to the threat of subjects trying to guess the intent of the study and altering their behavior. Further, we use the survey in OSS communities to understand whether or not openness helps organizations to accelerate the organization's internal development process. This introduces the risk of respondents guessing the hypothesis correctly and start responding from community's view point to be more open. Consequently, the respondents may completely ignore the organizational interest to affect the results, which can be used to convince software organizations to be open even when it is not desirable for an organization to be open. To mitigate this risk, all respondents were clearly instructed to answer the survey questions according to their associated organizational unit instead of OSS communities.

Reliability

The reliability deals with to what extent the data and the analysis are dependent on the specific researcher, and the ability to replicate the study.

Member checking. All three authors analyzed the extracted data from the studies included in the related work. Furthermore, the guidelines provided by Popay et al. [152] were used to make the synthesis process objective, reliable, transparent and repeatable.

Audit trail. The first author kept track of all the data sheets created from the studies and monitored data consistency and correctness during the synthesis iterations.

5 Theory formulation

Based on the synthesized empirical evidence, summarized above, we here present the *the theory of openness for software engineering tools in software organizations*, according to the theory-building framework proposed by Sjøberg et al. [169]. According to the framework, a theory consists of: 1) constructs, 2) propositions, and 3) explanation, outlined in the subsections that follow.

5.1 Defining constructs for the Theory of Openness for Tools

We derived four constructs from the visual representation of When and Why in Figure 4: Strategy, Trigger(s), Outcome(s), and Level of openness. These are abstracted during the data synthesis, as described in Figure 1. The synthesis also resulted in a further abstracted openness model, presented in Figure 5. Furthermore, the mapping of constructs from the attributes in openness model is shown in Table 7. The definitions of constructs are below.

Strategy refers to managerial decisions on **when** and **why** a software **organization** should only use or use and contribute to OSS tools communities (e.g., Jenkins, Gerrit etc.) . An example strategy is to engage software developers in OSS tools communities (A14) for knowledge building and exchange of innovative ideas to support innovation (A15, A16) in organizations' proprietary products. Furthermore, when the product loses it competitiveness (A5), it could become a candidate to become open source to create a community around it. Consequently, it helps SIPDOs to create a good reputation in the community and steer them towards their own needs (A9, A20).

Trigger(s) highlight(s) the actors involved in either decision-making of openness or its implementation(e.g., managers and software developers). The initial trigger might come from managers as a result of paradigm shift (e.g., switch from Windows to Linux) or from software developers due to complex continuous integration process (A1, A3). For example, **S2** showed a case where Sony Mobile

Table 7: Attributes (see Figure 4) mapping between constructs and proposition

ID (s)	Construct (s)	Proposition (s)
A9 [S1, S2, S3]	**Strategy**	**P4 – Degree of investment**
A16 [S1, S2, S3]		
A20 [S3]		
A5 [S2, S3]		**P5 – Requires management approval**
A12 [S1]		
A14 [S2, S3]		
A15 [S1, S3]		
A1 [S2]	**Triggers**	**P3 – Process and product Innovation**
A3 [S2, S3]		
A23 [S1]		**P4 – Degree of investment**
A19 [S1, S3]		**P5 – Requires management approval**
A2 [S2, S3]	**Outcomes**	**P1 – Reduce development cost**
A11 [S1, S3]		
A8 [S1, S2, S3]		**P2 – Shorten development time**
A13 [S1,S2]		
A18 [S1, S2, S3]		**P3 – Process and product innovation**
A22 [S1, S3]		
A4 [S1, S2, S3]	**Level of Openness**	**P1 – Reduce development cost**
A6 [S3]		
A10 [S1]		**P2 – Shorten development time**
A7 [S1, S2, S3]		**P3 – Process and product innovation**
A17 [S1, S2]		
A21 [S2, S3]		

switched from a proprietary tool ElectricCommander to an OSS tool Jenkins because it was easier to find solutions (A23) in OSS communities and adopt the tools according to the internal development work-flow (A19).

Outcome(s) deal(s) with the affect (implications) of adopting openness for tools in SIPDOs. Reduction in the development cost can be achieved by choosing open tools instead of licensed tools in SIPDOs for the internal software development (A2). It not only allows developers to stay up to date with their tool chain (A22), but also provides an opportunity for an organization's employees to interact with the developers in the tools communities (A11) and receive free new features and bugs corrections (A8). Open tools make the internal development environment more flexible since the developers themselves can enhance the tool to better suit their needs. Consequently, it can have an impact on the reduced development time which leads to shorter time-to-market (A13).

Level of openness entails the extent to which an organization should be open, which may be a property of both the development **process** and the **outcome**, according to Huizingh et al. [83]. For example, an openness of the development process for a SIPDO may include arranging a hackathon with tools community developer to allow internal developers to work together to implement new functionality (A7, A17). On the outcome level, an organization may choose to release the code to OSS communities instead of maintaining an internal fork for a commodity code (A4). Forking will only lead to more patching and code maintenance every time there is a new release from the community. Therefore, choosing the right level of openness may allow the organization to share the maintenance cost with the community to free up the internal developers time by receiving the latest patches from the community (A6).

5.2 Propositions for the Theory of Openness for Tools

The deduction of propositions can be traced back to constructs and attributes, see figure 4) and Table 7.

Proposition 1 (P1) Openness of tools provides opportunities to reduce development costs.

P1 relates to two constructs: 1) Level of openness and 2) Outcome(s). Openness of tools can lead to reduced development costs for multiple reasons. First, instead of "re-inventing the wheel" software organizations can choose the OSS tools already developed (A11, A2). Second, the costs of maintaining software can be shared with OSS communities (A4, A6) as showed by S2.

Proposition 2 (P2) Openness of tools provides opportunities to shorten the development time.

P2 relates to two constructs: 1) Level of openness and 2) Outcome(s). Openness of tools may lead to shorter development time (A13) due to the following reasons.

First, bugs related to OSS tools (e.g., Jenkins and Gerrit) could be fixed by the organization itself since they have access to the source code of the tools or they may receive fixes (A8) from community developers (S2). Second, software reuse (A10) and free features and bug fixes from communities enable software organizations to shorten the development time. S3 [App D] shows that contributors (57%) and non-contributors (59%) to OSS communities think that openness reduces development time.

Proposition 3 (P3) Openness of tools complements internal processes and product innovation.

P3 links to three constructs: 1) Level of openness, 2) Outcome(s), and 3) Trigger(s). S2 shows that the use of Gerrit and Jenkins communities in Sony Mobile's internal software development, leads to better internal development environment (e.g., continuous integration). The initial trigger of open tools came from the complex integration and building process (A3) and the shift from Windows to Linux (A1) As an outcome of this openness towards tools, it introduced a new fun way of working with tool communities for internal developers (A17, A21). Moreover, it enabled other units of Sony (e.g., Sony Entertainment) to replicate (inner source initiatives) the same development environment (A22), without having to pay for it. The better internal development process due to the openness towards Jenkins and Gerrit communities can be seen as an example of process innovation, which in-turns affects the improved competitiveness in products (A18).

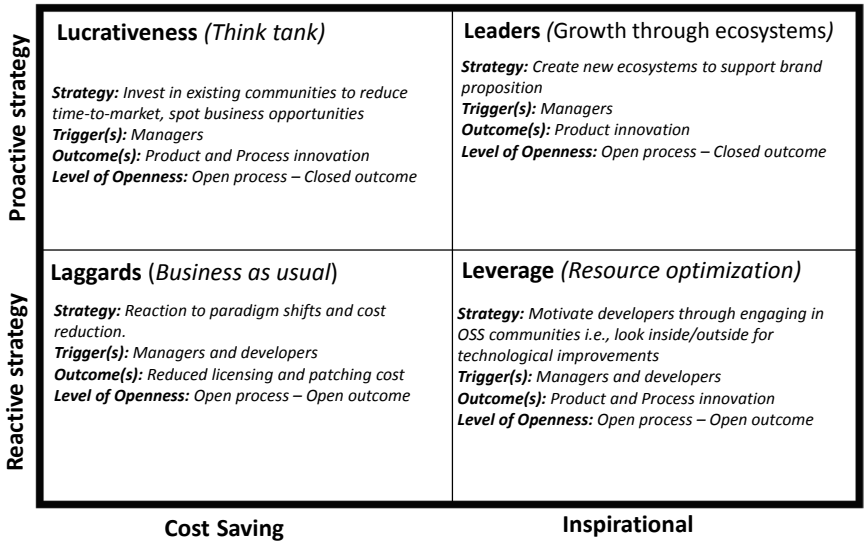
Proposition 4 (P4) The degree of investment in OSS communities has an affect on the outcome.

P4 relates to two constructs 1) Trigger(s) and 2) Strategy. SIPDOs may need to invest in OSS tools communities to build the reputation (A9), which helps organizations to gain influence and control over the communities (S2, S3). Consequently, an organization may have a better support external support for their internal product innovation (A16, A20).

Proposition 5 (P5) Introducing a proactive strategy, in relation to openness of tools, requires conscious management involvement.

P5 refers to two constructs 1) Trigger(s) 2) Strategies. The idea of acquiring the external knowledge (A19) to support internal innovation (A12, A15) may require organizations to move from a closed innovation model to an open innovation model. One such example is the creation of new open tools communities (A14), if the existing communities are not in-line with the organization's business model. This proposition is hinting that management may have to make proactive strategies to harness the power of internal employees, as well as the outside crowd in OSS communities, while still safeguarding the competitive edge (S3).

Figure 5: Model of Openness for Tools



5.3 Explanation of the Theory of Openness for Tools

The synthesized model, underpinning the theory of openness for software engineering tools in software organizations, is presented in Figure 5. The constructs of the theory, and instances thereof, are presented in **boldface** in the description below.

The theory explains four categories of organizations, represented by the quadrants in Figure 5. Each category has the different **levels of openness**, based on their strategies (**proactive** or **reactive**) in relation to goals (**cost saving** or **inspirational**) and associated with propositions (see Table 7). The theory presents four classifications of openness with their respective focus: 1. **Laggards** – Routine business, 2. **Leverage** – Resource optimization, 3. **Lucrativeness** – Acting as a think-tank, and 4. **Leaders** – Growth through ecosystems.

Laggards (Reactive strategy – Cost saving)

The underlying assumptions for the laggards is that they respond to paradigm shifts and all strategies are **reactive** in order to reduce the development **cost**, wanting to run **business as usual**. Organizations that position themselves in this quadrant understand that their software tools are not competitive and that alternative options are available, often as OSS. Studies indicate [22, 118, 120] that in the absence of

intellectual property rights, there may be greater chances of commutative advancements and reduction of the patching cost, by avoiding forking of OSS.

For example, the introduction of Git made IBM's ClearCase an expensive proposition for SIPDOs due to licensing costs. Forking an OSS tool for internal use may lead to patching costs (e.g., internal maintenance) as a result of new releases from the community. Therefore, managers and developers in organizations categorized as laggards should keep their development processes and outcomes **open** in order to receive all the latest updates from the community without patching it.

Leverage (Reactive strategy – Inspiration)

Organizations that are categorized in the leverage category, use external sources of innovation by **inspiring** their internal developers to participate in various OSS communities, prior to internal R&D work. The objective is to create synergy and synchronization between the organization's own processes and externally available ideas that could be incorporated in products, thereby **optimizing the resources** of the organization. This is a deliberate ploy from **managers** to absorb the external ideas and to make them fit for internal process. Furthermore, it not only adds to **product** and **process** innovation, but also **inspires** developers to participate in the discussion forums and exchange ideas to develop competence. The participation of organizations in communities like Opensource.com [192], Gerrit [137], and Acceptance test harness [139] are examples of open principles of catalyzing a community and engaging people through a common platform.

Lucrativeness (Proactive strategy – Cost saving)

Lucrativeness deals with investing in existing OSS communities to be able to influence and steer these communities in the same direction as the organizational interests, functioning as a **think-tank** for the organization. The objective is to support internal innovation and **reduce costs** by investing in OSS communities. The use of platforms helps organizations to reduce time-to-market. The key goal for the organization is to build the capability for its employees to make independent decisions, act quickly, take initiatives, and creatively solve problems. However, this goal requires employees' engagement, which need to be recognized by the managers. In order to make this work, **managers** need to ensure that processes are **open** for the exchange of ideas, but innovation outcomes are **closed** to achieve the competitive edge.

For example, CloudBees used Jenkins and added an extra layer of enterprise-grade security, scalability, manageability and expert-level support to improve the continuous integration in enterprise applications and started selling their product portfolio based on differentiation. Therefore, CloudBees utilized on Proactive – Cost saving strategy to not only reduce cost but also created a business opportunity to create revenue.

Leaders (Proactive strategy – Inspiration)

Leaders are organizations who look for emerging markets and breakout technologies by identifying the set of target areas, in order to **create growth through ecosystems**. These target areas are identified using lucrativeness. This is a top down exercise tied to strategic product planning lead by **top management**. The focus is on creating new communities and ecosystems that have the ability to disrupt business models.

For example, Sony created an ecosystem by making an Authoring Tools Framework (ATF) open source, which is used to make game development tools on Windows. It relieved PlayStation developers from contract barriers and made it quicker, easier and cheaper for developers to create development tools and game engines. By creating an ecosystem, Sony not only reduced the game development cost for its users but also strengthened the use of its PlayStation. The key is to safe-guard the competitive edge by keeping the processes **open**.

5.4 Definition of scope for the Theory of Openness for Tools

The scope defines the boundaries under which the theory of openness for software engineering tools in software organizations holds. We follow the advice of El Eman et al. [52], to "be sure to specify as much of the industrial context as possible. In particular, clearly define the entities, attributes and measures that are capturing the contextual information".

This information about the software organizations underpinning the theory, is addressed in Section 4.2 and reported in detail in F. It includes all software organizations, either only using or contributing as well to OSS communities, from which the empirical evidence is collected. F highlights the size, roles and OSS tools used by the software organizations that are adopting openness. Below we summarize the scope in terms of *technology, actors* and *software systems*.

Technology

Scope of validity: The use of OSS communities in the internal development of a software organization is compulsory. The organizations size may range from 1–50, 51–200 or $>$201 as mentioned in Figure 3.

Scope of interest: The knowledge extraction from OSS communities to facilitate organization's internal process, product innovation and reduced development time.

Actors

Scope of validity: Actors include software developers and managers. Software developers are employees associated with software organizations and also active

members of OSS communities.

Scope of interest: To engage internal developers from SIPDOs in OSS tools communities for process and product innovation.

Software systems

Scope of validity: Although the case study [S2] included in the synthesis process was performed at Sony Mobile with the focus on using and contributing to specific OSS tools communities, namely Jenkins and Gerrit, the survey includes software organizations with the wide range of OSS tools mentioned in Tables 1 and 2, the column Product type.

Scope of interest: Projects that involve the use of OSS tools.

5.5 Theory evaluation

In this section, we evaluated the Theory of Openness for Tools using the criteria proposed by Sjøberg [169]. The criteria are comprised of *testability, empirical support, explanatory power, parsimony, generality* and *utility*. For each criterion, we rated Theory of Openness on the scale of *low, moderate* or *high*.

Testability

The theory of openness for software engineering tools in software organizations presents understandable, consistent and unambitious constructs and propositions from the viewpoint of software developers and managers. Hypotheses are derived from the propositions since the scope conditions are clearly defined in the previous section. However, the adoption of OI using OSS needs to be understood by the readers in order to understand the theory.

There are a few studies conducted on the definition of openness in OI [39, 66, 114]. However, in this study, we choose the openness classification by Huizingh [83] since it was related to the opening of processes and outcomes, and the theory of openness has the level of openness as a construct (see Section 5.1).

We intend to validate the theory by conducting a workshop in multiple companies using OSS tools for their proprietary software product development. The workshop will be performed in two steps. In the first, the participants will be asked to select a tool from their own settings and categorize the constructs given to them using a Likert scale in relation to proactive/reactive strategy and cost/inspiration. Second, the discussion will be held to identify the differences in the participant's ratings and to verify the propositions. *We thus rate the testability of this theory as high*.

Empirical support

We identified three studies [131, 137, 176] highlighting the reduction in development cost **(P1)** and two studies [130, 137] pointed out the reduced time-to-market for software organizations due to the use of OI **(P2)**. It must be mentioned that the reduced time-to-market refers to shorter development time only, and not the marketing. Furthermore, eight studies [78, 130, 131, 137, 145, 168, 173, 176] underpin the proposition on process and product innovation **(P3)** as a result of Open Innovation. Table 8 shows that **P1** and **P3** has relatively stronger empirical evidence support compared to **P2**, **P4** and **P5**. The interview data from studies [130, 137, 176] suggest that OI helped organizations to reduce time-to-market (**P2**), improve innovation capacity and speed (**P3**), and reduce the development cost (**P1**).

Henkel et al. [78] and Parida et al. [145] conducted a case study on drivers for embedded systems and a survey in 252 IT organizations, respectively. Both studies conclude that OI needs to put on the "radar screen of the top management" (**P5**) in order to increase competitiveness, better and faster development releases (**P3**). In relation to **P4**, Parida et al. [145] concluded that outside-in OI is positively associated with the organization's radical innovation performance (products new to the world) and incremental innovation (products new to the firm) performance. Henkel et al. [78] also concluded that disruptive innovation challenges managers to look for customers' demands and competitors implementing open models of innovation. However, it raises the question of making new proactive strategies (**P5**) to respond to disruptive innovation challenge. For example, disruptive innovation may suggest to make a separate organizational unit tasked with this challenge as highlighted in **S2**.

Morgan et al. [131] presented a study of 13 managers in the software sector in Europe and examined how their perceptions of the benefits and drawbacks of OSS using OI affected their decision to use OI in their organizations. The study shows benefits in terms of increased collaboration, escape from vendor lock-in and encouraging innovation, permit companies to team up with other companies, customers, and universities to overcome certain adoption factors, like technological complexity and facilitate product development.

It should be noted that neither of the aforementioned studies state whether benefits of openness were as a result of a reactive or proactive strategy. Furthermore, the scope of the studies are not limited to tools only, and therefore addressed related to the internal validity threats (Section 4.3). The proposed theory is supported or partly supported by the studies mentioned in Table 8. *Therefore, we consider the empirical support for this theory to be moderate*.

Explanatory power

The proposed theory is defined from the abstractions of the synthesis process [26,126,198]. The theory highlights the openness for software engineering tools in software organizations in relation to reduced development cost, reduced develop-

Table 8: Empirical evidence to support propositions. **P1**, **P2**, **P3**, **P4** and **P5** represent propositions mentioned in Section 5.2

Proposition	Evidence	Summary
P1 – Reduce development cost	[131, 137, 176]	E.g., reduced licensing cost, patching cost
P2 – Shorten development time	[130, 137]	E.g., reuse of OSS, outsourcing of software testing and bug fixing and maintenance
P3 – Process and product innovation	[78, 130, 131, 137] [145, 168, 173, 176]	E.g., faster testing and debugging, better work-flow management for developers, free-up time, incremental innovation, better continuous integration
P4 – Degree of investment	[145]	Positively associated with radical and incremental innovation
P5 – Requires management approval	[78]	E.g., creation of new departments working with innovation through openness

.

ment time and increased product and process innovation. Furthermore, the theory highlights the motivations of why (conscious management strategy) organizations should become open. *Therefore, we consider the explanatory power of this theory as moderate.*

Parsimony

Parsimony deals with how economically a theory has been constructed in relation to a number of constructs and propositions. The proposed theory is derived from the rich amount of qualitative data extracted from the mapping study (**S1**), case study (**S2**) and survey (**S3**). However, the formulated theory of openness, uses a high level of abstraction, with clear explanation of transitions between theoretical steps. The number of constructs and propositions related to OSS tools were kept low. *Therefore, we consider the parsimony of the theory as moderate.*

Generality

The scope of the proposed theory is limited to software organizations using OSS tools in the internal software development, which makes *the generality of the theory low*.

Utility

From a software organization's perspective, the proposed theory can be utilized for choosing the right level of openness while working with OSS communities. Most software organizations have to work with OSS communities therefore, *we see utility of the theory high*.

6 Conclusion and future work

This paper presents a theory of openness for software engineering tools in software organizations, that helps SIPDOs to understand why and how organizations should capitalize the potential that OSS communities bring to leverage the internal R&D. In relation to the first research goal defined in section 3.1, the presented theory highlights which type of organizations are involved in embracing openness, and what are the possible challenges (intellectual property rights, complete test coverage, the absence of business strategy etc.) faced by these organizations in relation to openness. Furthermore, this paper highlights some of the potential metrics that could be used to measure innovation as a consequence of openness (see Figure 2).

However, there are some drivers that need to be understood by SIPDOs to be successful and open at the same time. These drivers entail cost, time-to-market, speed, process and product innovation. Based on the aforementioned objectives, the theory fulfills the second research goal by introducing the level of openness and associated strategies that should be adapted, based on objectives (e.g., cost reduction, reduced development time etc.). Further, it helps positioning an organization with respect to its strategy to act as Laggard, Leverage, Lucrativeness or Leader (see Figure 5).

Future work includes validating the theory with organizations using OSS tools in their product development to generalize the findings across a broader range of SIPDOs. The validation could possibly discover more reasons for openness depending upon the organizational context. Furthermore, the validation may lead to guidelines for managers depending upon the objectives to achieve using openness.

APPENDIX D

SURVEY DESIGN

This appendix presents the survey design details as well as the outcomes. The questionnaire[1] was divided into two branches, namely contributors and non contributors. The survey questionnaire was distributed among 500 employees working for software-intensive organizations either using Gerrit, Jenkins and Git communities in their development or also, contributing to those communities. The questions were divided into the following categories.

1. Demographics
2. Involvement in OI using OSS projects
3. Operationalization of OI in software engineering
4. Quality assurance

First, five questions were common for both contributors and non-contributors related to the demographics e.g., organization, working experience with OSS, job title, work responsibilities etc. We received 57 responses including contributors and non contributors to the communities.

We extracted the email list of Jenkins, Gerrit and Git communities from GitHub and distributed the survey among all those contributors and non contributors having organizational affiliations in their email addresses. Contributors refer to all those employees who are contributing source code, documentation, test cases etc. to Jenkins Gerrit and Git. On the other hand, users refer to employees only using OSS software e.g, the use of Jenkins and Gerrit, in the development of their organization's products or services.

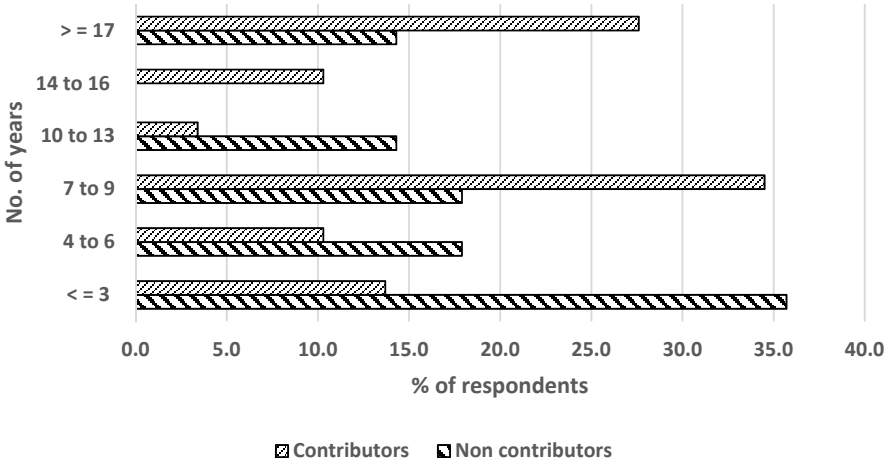
The study aimed at responses from the perspective of employees in the context of the nearest organizational unit, using or contributing to OSS communities (Jenkins, Gerrit and Git). The underlying assumption was that the managers may know the overall strategy of the organization, but software developers/testers may only have the local knowledge of their unit. Furthermore, during the pilot survey,

[1] http://fileadmin.cs.lth.se/cs/Personal/Hussan_Munir/Surveyform.pdf

respondents highlighted that they could only respond based on their organizational unit since they do not know the OSS contribution strategy of the whole organization, and it may vary from one unit of another unit.

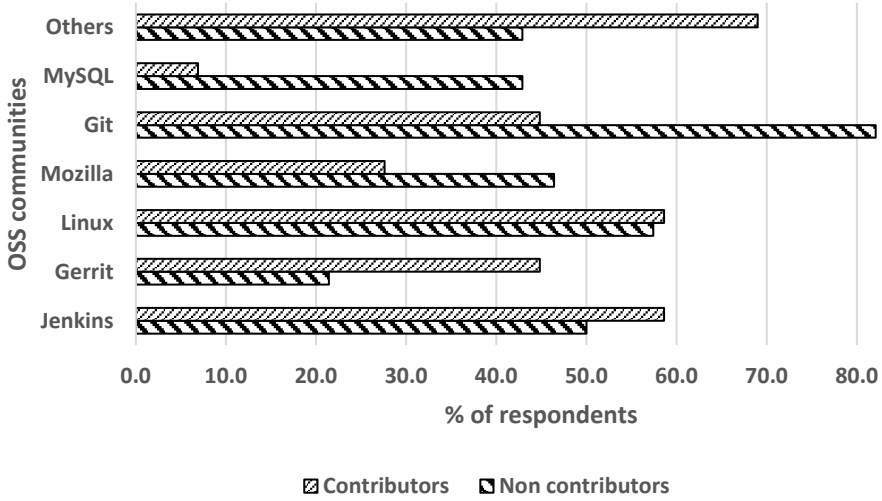
1 Demographics

Figure 1: Distribution of survey respondents' experience (%)



Of the survey respondents 61% were software developers, followed by managers (14%), system architects (11%) and testers (9%) and CEOs (4%). Figure 1 represents the number of years of experience of the respondents and Figure 2 shows the association of respondents to OSS communities. 68% of the non contributors have 7 or more years of experience using OSS communities in their respective organizations. On the other hand, 79% of contributors have 7 or more years of using and contributing to OSS communities. It must be mentioned that survey was initially distributed among Gerrit, Jenkins and Git communities to reach to desired sample population, but it turns out that respondents are working for multiple OSS communities. Therefore, Figure 2 shows that a respondent may be associated with multiple OSS communities simultaneously.

Figure 2: Distribution of respondents' association with OSS communities (%)



APPENDIX E

WHY GET ORGANIZATIONS INVOLVED IN OI USING OSS?

Why refers to the factors considered by SIPDOs adopting OI using OSS. OI adoption is not only driven top down, but also bottom up since the majority of the respondents (72%) in the survey are either tasked by the management to take OI initiatives or volunteered (62%) to do so. Some respondents (21%) also see it as a fun way of working with communities as a part of their daily work. The others category (38%) revealed more factors considered by software organizations before opening up. These factors entail the cost of maintaining forks of OSS code, good discussions forums for exchanging innovative ideas, easier to find a solution to an issue, saves own development cost, create the base ecosystem to innovate and create innovative products that can delivered to the clients, cheaper than in-house development, to motivate/educate engineers in the organization, reduced maintenance cost, continuous development and customer satisfaction.

The survey results also highlighted that organizations see the need to build OSS communities to attract external knowledge into the organization. It is clear from the responses that software organizations make a project open if it is a non-competitive tool or a products that is not a direct source of revenue anymore. The respondents highlighted more contribution strategies (see Figure 1): 1) employees working with OSS communities encourage organizations to become open and to gain good reputation in the community, 2) when there are patches involved with the bug correction and licenses that mandate organizations to reveal code, 3) when the contributions are not only specific to the organization and have value for the community as well, 4) competitors' pressure is also one of the main motivations for adopting Open Innovation and acquiring new features in OSS tools such as Jenkins, Gerrit etc.

Figure 1 presents the differences between contributors and non contributors in incentives to contribute to OSS communities. The incentives are rated by respondents on a likert scale from *less important* to *very important*. The majority of contributors and non contributors agreed in highlighting the importance of getting the latest patches from OSS communities (59% and 57%, respectively). However,

Figure 1: Motivating factors to contribute to OSS communities

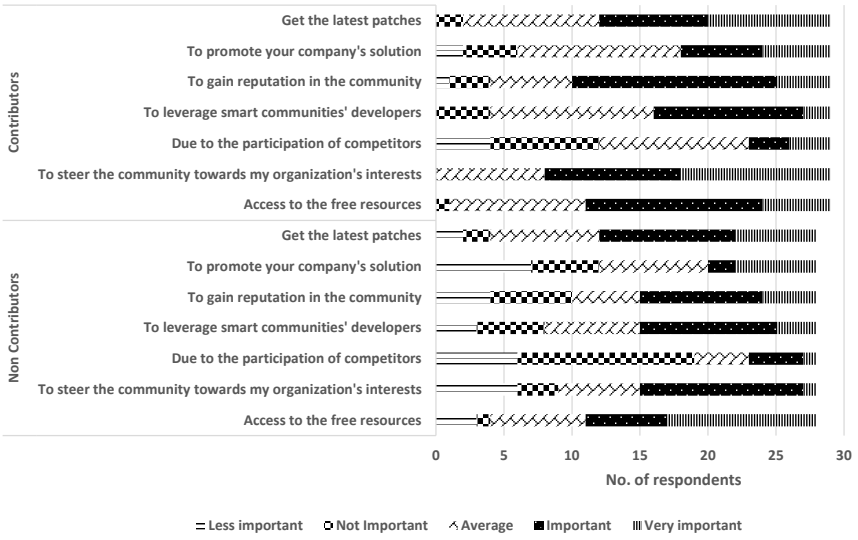


Figure 2: Factors gained by companies after contributing to OSS communities

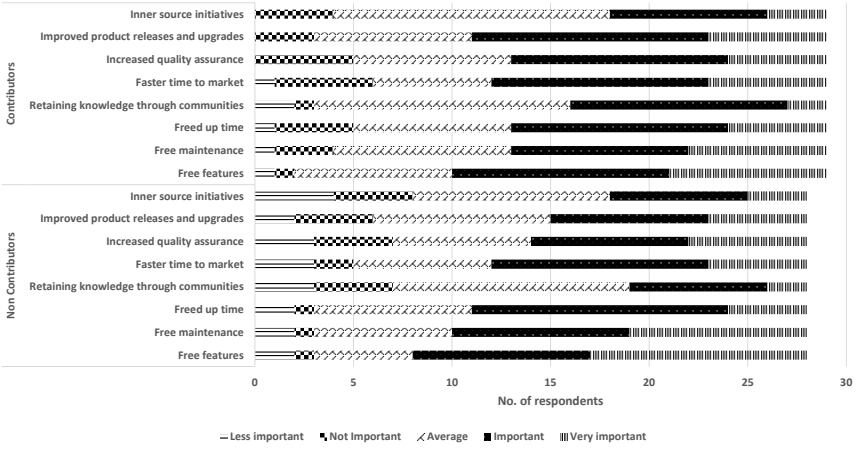


Table 1: Time spent on OI

Time spent	**%**
Less than 5 hours	49
6 – 10 hours	14
10 – 15 hours	3
16 – 20 hours	0
21 – 25 hours	3
Full time	31

there is a difference of opinion between contributors and non contributors when comparing the reputation gain in the community and steering the communities towards organizational interests. 72% of of contributors think that gaining a good reputation is important to steer the community towards their organizational interests. At the same time, only 46% of non contributors think that is important to gain reputation in the community in order to steer the community towards organizational interests.

Figure 2 shows the potential gains reported by contributors and non contributors by involving themselves in OSS communities. Both contributors (57%) and non contributors (59%) agreed that the use of OSS communities reduces time to market for the development of products as it frees up developer time. However, more contributors 62% think that using and contributing to OSS communities resulted in improved product releases and upgrades as oppose to 47% of non contributors.

1 Operationalization of Open Innovation in software engineering

Table 1 shows that 63% of the respondents spent less than ten hours per week working with OI. At the same time, 31% respondents in the organizations spent their full time with OI by working with the OSS communities, which indicates that software organizations are realizing the importance of OI to extract and assimilate the external knowledge from communities in the organization's product development.

Regarding contributions to OSS communities, engineers (69%), middle (55%) and top level management (52%) are involved in deciding whether or not to contribute organization's internal source code to OSS communities. However, there is an extra layer of legal managers (45%) helping organizations to deal with licensing, intellectual property rights and patent infringements.

2 Quality assurance

Of the respondents 59% use OSS tools e.g., JIRA while 31% use Google's issues tracker for bug reporting. However, bugs are reported to communities by email (28%) and weekly/monthly meetings (21%) among the contributors. One possible explanation of reporting bugs through email or meetings is that those bug could be relevant to the organizations only and contributing it to the community would not give any value.

Among the the respondents 45% reveal more than 75% of their organization's source code to the communities. A plausible explanation of this could be that the source code is not seen as a competitive advantage, or to build a community around the non competitive tools such as Jenkins or Gerrit. On the other hand, 52% of the respondents choose to reveal less than 25% of the code to the communities, due to the risk of losing the intellectual property rights or due to lack of understanding of the OSS culture in the organization.

Out of the respondents 79% mentioned that bug fixing is prioritized based on the needs of organizations. It was also confirmed by S2 that if a bug is a make or break for the organization's development process then the organization choose to fix it. Otherwise, bugs are pushed into the Google's issue tracker for someone else to fix it. Furthermore, 38% of the respondents think the bugs are prioritized based on the communities needs as well in order to gain good reputation in the community.

APPENDIX F

WHO – ORGANIZATIONS INVOLVED IN OPEN INNOVATION

Table 1: Non contributors

Organization names	**Product type**	**People:(Size, Roles, experience)**	**OSS Tools**
Perten Instruments AB	Mechanical or Industrial Engineering Web based monitoring and analysis applications	51-200 employees, Present in over 100 countries System architect, Software developer 7-9 years	Linux, Git, jQuery, knockout, bootstrap, jcanvas, requirejs.net, Jenkins, Mozilla Firefox
Sysart, Finland	Computer Software	51-200 employees, Software development, System architect, 4-6 years	Jenkins, Gerrit, Linux, Mozilla Firefox, Git, MySQL, nginx, openjdk, apache httpd, postgresql
Vnomics, Inc.	Transportation	51-200 employees, Software developer, 7-9 years	Jenkins, Linux, Mozilla Firefox, Git, MySQL, Mercurial, Java
Henan University, China	Academics	12500 students, Software developer, $<$ 17 years	Jenkins;Linux;Mozilla Firefox;Git;ant, libreoffice
Polytechinc of Milan, Italy	Academics	40,000 students, Manager, Teacher, < 17 more	Mozilla Firefox;Git
Spirosoft	Expertise in both embedded systems and desktop computing.	1-10 employees, Software tester, Write test-cases, Execution, $>$ 3	Mozilla Firefox, Git
Datenc	Cloud computing Cloud Solutions Data storage solutions	1-10 employees, Software Designing, Development and Management, System Architect $>$ 3	Linux, MySQL
GRID Systems Pakistan	E-commerce services, Software Development and teamwork. Clinic Software	1-10 employees, Software Designing, Development and Management, Software tester 4-6 years	Linux, MySQL
OSSE, USA	Mechanical or Industrial Engineering	51-200 employees, Software developer, 7-9 years	Git

Meltwater	Information Technology and Services	1001-5000 employees, Software developer, NA	Jenkins, Linux, Git, MySQL
Agilecrm and Hyd	Customer Relationship Management system	51-200 employees, Software tester, 4-6 years	Linux, Mozilla Firefox, Git
XLN Audio AB, Stockholm	Music software	11-50 employees, software developer, system arcchitect, 10-13 years	Git, JUCE, juce, MySQL
Confiz, Pakistan	Specialties Portal Applications, Mobility, Cloud Computing, Content Management, Cross Platform Frameworks, Android applications	201-500 employees, Principal Software Engineer, Software tester, Team leader, 7-9 years	Linux, Git, MySQL Jenkins, Mozilla Firefox, Git, MySQL
Follow-up System, Stockholm	Information Technology and Services	11-50 employees, Software developer, $>$ 3 years	Jenkins, Linux, Mozilla Firefox, Git, MySQL
Saudi Telecom organization	Telecommunications sector	10,001+ employees, System architect, 10-13 years	Jenkins, Gerrit, Git, MySQL

Table 2: Contributors

Organization names	**Product type**	**People:(Size, Roles, experience)**	**OSS Tools**
Ericsson	Telecommunications Industry	More than 100,000 employee, More than 37,000 patents, 40% of mobile calls are made through our systems 180 countries, Manager, Software developer, $<$ 10 years	Jenkins, Gerrit, Linux, Git, Sonarqube, Casandra
SAP	Enterprise application software	10,001+ employees, Software developer, 7-9 years	Jenkins, Gerrit, Linux, Git, BUILD, Eclipse
Google, USA	Specialties include search, ads, mobile, android, online video, apps, machine learning, virtual reality	10,001+ employees, Software Engineer, 7-9 years	Gerrit, Linux, Git
Axis, Sweden	Specialties include Network video, IP Video surveillance, Security camera, CCTV, IP camera Industry Computer Networking	1001-5000 employees, over 80,000 partners, Software developer $<$ 17 years	Jenkins, Gerrit, Linux
Sony Mobile Sweden		10,001+ employees Software developer, Manager, System architect $<$ 7 years	Jenkins, Gerrit, Linux, Mozilla Firefox, Git, Jgit, eclipse, repo, gitlab, android, Jgit, GitLab, Repo, Chromium, mainly Android
Mozilla Vancouver	Specialties include browser, internet, software, mobile, web apps, OS, identity	501-1000 employees, Software developer testing, $<$17 or more	Mozilla Firefox, mercurial
Intel Lund	Specialties includes semiconductor design and manufacturing	10,001+ employees, 63 countries, Manager Managing a team of SW engineers and managing the Intel Lund site, Manager $<$ 17 years	Linux

Thales UK	Specialties include Aerospace, Defence, Security, Space, Transportation, Cybersecurity	10,001+ employees, Software developer support (installation/configuration) of all software/system development tools, linux admin, virtual data center admin, 14-16 years	Git
ESO, Germany	Specialties include astronomical research technology and producing scientific libraries under GPL	501-1000 employees, Software developer Scientific software development and maintenance, regression testing, $<$ 14 years,	Junit, Selenium, Also customised tools to interface with C/C++ and Python code.
SmartKompare.com Dhaka	Specialties includes Financial Technology, Information, Comparison	11-50 employees, CEO Managing team, leading tech team, We develop modules etc , $>$ 3 years	Drupal
Red Hat Spain	Specialties in cloud computing, hybrid cloud management, Linux, open source, virtualization, storage, middleware, containers, mobile, OpenStack	5001-10,000 employees, Continuous Integration Engineer Design,$<$ 17 years	Jenkins, Gerrit, Linux, Git, MySQL, Fedora, openstack, gnome, kde, gimp
Flownative Gmbh, Germany	Flownative helps organisations creating first-class Neos websites and sophisticated Flow applications.	1-5 employees, Developer and CEO Software development and testing, $>$ 3 years	Neos, Doctrine & other (mostly PHP) projects
A National Laboratory, USA	Research and development	Software developer, $<$ 17 years	Jenkins, Linux, Mozilla Firefox, Boost C++
Garmin	Specialties include Consumer Electronics, Worldwide leader in navigation & communication products, Products in aviation, marine, fitness, outdoor, & automotive products	10,001+ employees, Software developer Lead a team, 7-9 years	Jenkins, Gerrit, Linux, Git

Qvantel, Karlslrona Sweden	Specialties include Cloud based Business Support Systems (BSS), Software Development Services, Offshore Software Development for small/mid size software vendors, Business Process Management (BPM) Services	201-500 employees, Auto execution of integration test, Build failure/Success, Regression testing, 4-6 years	Jenkins, Linux, Mozilla Firefox, Git, Bigdata, Stash
Confiz	Specialties include Portal Applications, Mobility, Cloud Computing, Content Management Touch, Quality Engineering, .NET Frameworks, JS Frameworks, Windows 8, Android applications	201-500 employees, Software developer, Project documentation, Requirement Analysis and writes Automation Script, 7-9 years	Jenkins, Linux, Git, MySQL Jenkins, Mozilla Firefox, Git, MySQL
Yatta Solutions Germany	Specialties include Eclipse, software development, software architecture, software engineering, UML	11-50 employees, Software developer, 7-9 years	Eclipse

1 Example of raw data collected from S1, S2 and S3

Table 3: Selected raw data from S1, S2, and S3. Please note that it is only a small part of raw data collected from the three studies to show an example.

Facets	S1	S2	S3
Organizations using OSS tools e.g., Jenkins, Gerrit, Git	Nokia, IBM, RdHat, HP	Sony, Google, Ericsson, HP, SAP, Intel, Latombe, Black-Build, Redhat, Codeaurora, Quelltextlich	Perten, Sony, Intel, XLN Audio AB, Confiz
Strategies used by SIPDOs for openness	Accessing and extending the resource-base of the firm, Aligning the firm's strategy with the community, Integrating and sharing results, Selective revealing	Difficult to keep up with the community's pace due to lack of resources, More patching as result of different direction than the community, Access to pragmatic software development workforce, To influence or steer community towards company's business model	To build a community around the project, When the product looses competitiveness, When the product is the main source of revenue, Because your competitors are making their projects open source, Non competitive tools only

Factors considered by software organization for openness	Knowledge building and exchange, Platform and software reuse, Innovation support, Time to market, cost, maintenance Culture change	To move from Windows to Linux, Proprietary solutions to OSS solutions, Ease off the complex integration and building process,	Fun way of working, Tasked by management, cost of maintaining forks of OSS code, good discussions forums for exchanging innovative ideas , Influence and reputation
Challenges faced by organizations to adopt OI	Business strategy (i.e unclear contribution strategy), Strategic OI barriers (i.e. lack of expertise), Governance (i.e. giving up control)	Legal issues, Intellectual property rights, Lack of OSS culture, lack of understanding, Risk of losing competitive advantage	Time, Protecting intellectual property Dependency towards an open source community for support, It steal's focus from our main tasks, I guess the potential has not been identified by the managers., End user requirements are highest priority, Business secrets/policies, Lack of resources, Lack of developer time

2 Rigor and relevance criteria

2.1 Rigor

Context(C)

1. **Strong description:** The context is described to the extent where it becomes comparable to other settings [85]. 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 [85]. 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.

2.2 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.

CHAPTER IV

MOTIVATING THE CONTRIBUTIONS: AN OPEN INNOVATION PERSPECTIVE ON WHAT TO SHARE AS OPEN SOURCE SOFTWARE

Abstract

Open Source Software (OSS) ecosystems have reshaped the ways how software-intensive firms develop products and deliver value to customers. However, firms still need support for strategic product planning in terms of what to develop internally and what to share as OSS. Existing models accurately capture commoditization in software business, but lack operational support to decide what contribution strategy to employ in terms of what and when to contribute. This study proposes a Contribution Acceptance Process (CAP) model from which firms can adopt contribution strategies that align with product strategies and planning. In a design science influenced case study executed at Sony Mobile, the CAP model was iteratively developed in close collaboration with the firm's practitioners. The CAP model helps classify artifacts according to business impact and control complexity so firms may estimate and plan whether an artifact should be contributed or not. Further, an information meta-model is proposed that helps operationalize the CAP model at the organization. The CAP model provides an operational OI perspective on what firms involved in OSS ecosystems should share, by helping them motivate contributions through the creation of contribution strategies. The goal is to help maximize return on investment and sustain needed influence in OSS ecosystems.

1 Introduction

Open Innovation (OI) has attracted scholarly interest from a wide range of disciplines since its introduction [186], but remains generally unexplored in software engineering [141]. A notable exception is that of Open Source Software (OSS) ecosystems [87, 185, 188]. Directly or indirectly adopting OSS as part of a firm's business model [32] may help the firm to accelerate its internal innovation process [31]. One reason for this lies in the access to an external workforce, which may imply that costs can be reduced due to lower internal maintenance and higher product quality, as well as a faster time-to-market [176, 180]. A further potential benefit is the inflow of features from the OSS ecosystem. This phenomenon is explained by Joy's law as *"no matter who you are, not all smart people work for you"*.

From an industry perspective, these benefits are highlighted in a recent study of 489 projects from European organizations that showed projects of organizations involving OI achieved a better financial return on investment compared to organizations that did not involve OI [48]. Further, two other studies [109, 137] have shown that organizations with more sources of external knowledge achieved better product and process innovation for organization's proprietary products. Moreover, a recent survey study [28] in 125 large firms of EU and US showed that 78% of organizations in the survey are practicing OI and neither of them has abandoned it since the introduction of OI in the organization. This intense practicing of OI also leads 82% of the organizations to increase management support for it and 53% of the organizations to designate more than 5 employees working full-time with OI. Moreover, the evidence suggests that 61% of the organizations have increased the financial investment and 22% have increased the financial investment by 50% in OI.

To better realize the potential benefits of OI resulting from participation in OSS ecosystems, firms need to establish synchronization mechanisms between their product strategy and product planning [62], and how they participate in the ecosystems and position themselves in the ecosystem governance structures [10, 141, 173, 194]. This primarily concerns firms that either base their products on OSS or employ OSS as part of their sourcing strategy. To achieve this synchronization, these firms need to enrich their product planning and definition activities with a strategic perspective that involves what to keep closed and what to contribute as OSS. We label this type of synchronization as *strategic product planning* in OI. *Contribution strategies* [194], i.e., guidelines that explain *what* should be contributed, and *when* play a vital role here. A common strategy is to contribute parts considered as a commodity while keeping differentiating parts closed [76, 185]. The timing aspect is critical as functionality sooner or later will pass over from being differentiating to commodity due to a constantly progressing technology life-cycle [120]. This strategy is further emphasized by existing commoditization models [22, 120]. However, these models are not designed with active OSS ecosystem participation

in mind and lack support for strategic product planning and contribution strategies.

In this paper, we occupy this research gap by presenting a Contribution Acceptance Process (CAP) model. The model was developed in close collaboration with Sony Mobile. Sony Mobile is actively involved in a number of OSS ecosystem, both in regard to their products features and their internal development infrastructure[1]. With the consideration of OSS as an external asset, the CAP model is based on the Kraljic's portfolio purchasing model which helps firms analyze risk and maximize profit when sourcing material for their product manufacturing [104]. The original model is adapted through an extensive investigation of Sony Mobile's contribution processes and policies, and designed to support firms' strategic product planning. More specifically, the model helps firms to create contribution strategies for their products and software artifacts such as features and components. Hence, the CAP model is an important step for firms that use OSS ecosystems in their product development and want to gain or increase the OI benefits, such as increased innovation and reduced time-to-market. Moreover, we help firms to operationalize the CAP model by proposing an information meta-model. The meta-model is an information support that should be integrated into the requirements management infrastructure and enables contribution strategies to be communicated and followed up on a software artifact-level throughout a firm's development organization. As a first validation outside of Sony Mobile, the CAP model was presented to and applied in three case firms. This provided understanding of the model's generalizability, and also input to future design cycles.

The rest of the paper is structured as follows: In section 2, we position our study with related work and further motivate the underlying research gap. This is followed by section 3 in which we describe the research design of our study, its threats to validity and strategies used to minimize these threats. In section 4 we present our CAP model and in section 5 we present an information meta-model for how contribution decisions may be traced. In section 6, we present an example of how the CAP model and meta-model may be used together inside Sony Mobile. In section 7 we present findings from three exploratory case studies outside Sony Mobile where we focused on early validation the CAP model's applicability and usability. Finally, in section 8 we discuss the CAP model in relation to related work, and specific considerations, while we summarize our study in section 9.

2 Related work

Below we describe the context of our research with respect to how software engineering and OSS fits into the context of OI. Further, we give a background on contribution strategies and commoditization models. Moreover, we provide a background of the sourcing model on which the CAP model is based. We than provide

[1] http://developer.sonymobile.com/knowledge-base/open-source/

an overview on what we label as strategic product planning, as well as on software artifacts, and conclude by describing the research gap, that this study aims to fill.

2.1 Open Innovation in Software Engineering

OI is commonly explained by a funnel model [31] representing a firm's R&D process, see Fig. 1. The funnel (1) is permeable, meaning that the firm can interact with the open environment surrounding it. This conceptualization fits onto many contexts, e.g., a firm that takes part in a joint-venture or start-up acquisition. In our case, we focus on ecosystems (2) and specifically those based on OSS [64, 87]. An OSS ecosystem consists of the focal firm along with other actors who jointly see to the development and maintenance of an OSS project, which may be seen as the technological platform underpinning the relationships between the actors [90, 123]. In the context of this study, the focal firm represented by the OI funnel is Sony Mobile and their internal software development process. The OSS ecosystem could, for example, be represented by that surrounding the Android Open Source Project[2] (AOSP). The interactions between the focal firm and the ecosystem (see Fig. 1) are represented by the arrows going in and out and can be further characterized as knowledge exchange between the firm and the OSS ecosystem (e.g., Sony Mobile and AOSP). Examples of transactions can include software artifacts (e.g., bug fixes, features, plug-ins, or complete projects), but also opinions, knowledge, and support that could affect any step of the internal or external development.

The interactions (3) may be bi-directional in the sense that they can go into the development process from the open environment (*outside-in*), or from the development process out to the open environment (*inside-out*). *Coupled innovation* [55] happens when outside-in and inside-out transactions occurs together (i.e., consumption of and contribution to OSS). This may be expected in co-development between a firm and other ecosystem participants in regard to specific functionality (e.g., Sony Mobile's developer toolkits[3]).

How firms choose to work with and leverage these interactions with OSS ecosystems impact how they will realize the potential benefits of OI, such as increased innovation, shorter time-to-market, and better resource allocation [176, 180]. The CAP model presented in this paper provides operational and decision-making guidelines for these firms in terms what they should contribute to and source of from the OSS ecosystems. I.e., how they should interact with the open environment in an inside-out, outside-in, or coupled direction. Hence, what the CAP model brings in terms of novelty is an operational OI perspective on what firms involved in OSS ecosystems should share, by helping firms motivate the contributions through the creation of tailored contribution strategies.

[2] https://source.android.com/
[3] https://github.com/sonyxperiadev

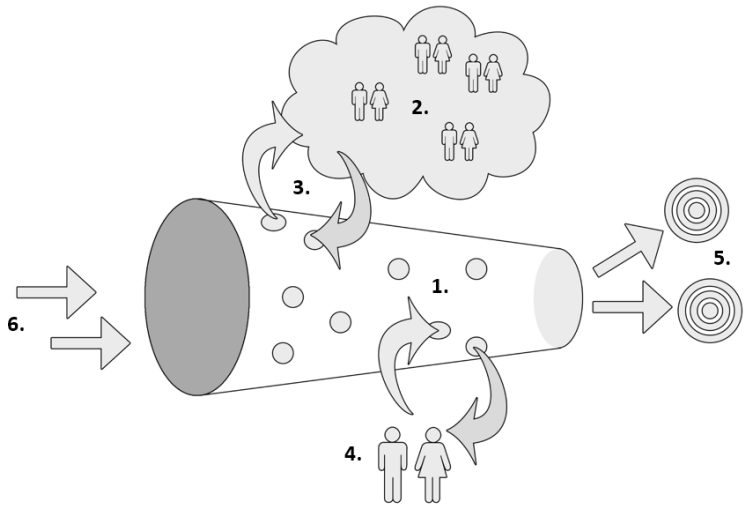


Figure 1: The OI model illustrated with interactions between the firm (1) and its external collaborations (2,4). Adopted from Chesbrough [31].

2.2 Contribution Strategies in Open Source Software Ecosystem

Wnuk et al. [194] define a contribution strategy as a managerial practice that helps to decide what to contribute as OSS, and when. To know what to contribute, it is important for firms to understand how they participate in various OSS ecosystems in regards to their business model and product strategy from an OI perspective. Dahlander & Magnusson [40] describe how a firm may access the OSS ecosystems in order to extend its resource base and align its product strategy with ecosystems' strategies. In another study, Dahlander & Magnusson [41] describe how a firm can adapt its relationships with the OSS ecosystems based on how much influence the firm needs, e.g., by openly contributing back to the OSS ecosystem, or by keeping new features internal. To build and regulate these relationships, a firm can apply different revealing strategies in this regard: differentiating parts are kept internal while commodity parts are contributed [76, 185]. Further, licenses may be used so that the technology can be disclosed under conditions where control is still maintained [185]. Depending on the revealing strategy the level of openness may vary from completely open, partly transparent conditions [32], to completely closed. As highlighted by Jansen et al. [89], the openness of a firm should be considered as a continuum rather than a binary choice.

2.3 Commoditization Models

With commoditization models, we refer to models that describe a software artifact's value depreciation [97] and how it moves between a differential to a commodity state, i.e., to what extent the artifact is considered to help distinguish the focal firm's product offering relative to its competitors. Such models can help firms better understand what they should contribute to OSS ecosystems, and when, i.e., provide a base to design contribution strategies [194]. Van der Linden et al. [120] stressed that efficient software development should focus *"... on producing only the differentiating parts"* and that *"... preferably, firms acquire the commodity software elsewhere, through a distributed development and external software such as [commercial software] or OSS"*. Firms should hence set the differentiating value of a software artifact in relation to how it should be developed, or even if it should be acquired. Commoditization is also related to the product's life-cycle and, is more often experienced towards the end of the life cycle [101].

Van der Linden et al. [120] present a commoditization model that highlights how commoditization is a continuous and inevitable process for all software artifacts. Therefore, firms should consider whether the software or technology should be developed, acquired, or kept internally, shared with other firms, or made completely open (e.g., as OSS) [11]. Ideally, differentiating software or technology should be kept internal, but as their life-cycle progresses their value depreciates and they should be made open. This is particularly relevant for software artifacts

that have an enabling role for cross-value creation, data collection or support value creation when combined with other parts of the offering, e.g., an artifact that collects and analyzes anonymous customer data that could be offered as business intelligence to customers [97]. Bosch [22] presents a similar commoditization model, which classifies the software into three layers and describes how a software's functionality moves from an early development stage as experimental and innovative, to a more mature stage where it provides special value to customers and advantage towards competition, then finally transitioning to stage where it is considered as commodity, hence it *"... no longer adds any real value"* [22].

A challenge identified by both van der Linden et al. [120] and Bosch [22] is the risk of losing Intellectual property rights (IPR) to competitors, a challenge that has also been highlighted in numerous other studies [76, 77, 188, 194]. By not contributing software and technology that are considered differentiating, firms can avoid the risk of giving away its added value to competitors. However, both van der Linden et al. [120] and Bosch [22] highlight how the acquisition of the commodity functionality may help firms to reduce the development and maintenance cost, and potentially shorten time-to-market. Instead, they can shift internal focus to differential features and better-justified R&D activities [120].

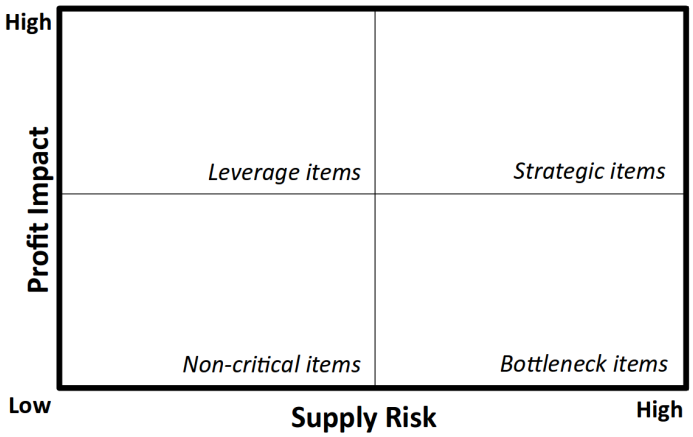
2.4 The Kraljic Portfolio Purchasing Model

From the software product planning perspective, sourcing refers to decisions of what parts of the software that should be developed internally or acquired externally, from where and how [101], and is an important part of a firm's product strategy [62]. A recent literature review of software component decision-making making lists four sourcing strategies: in-house, outsourcing, COTS and OSS and brings supporting evidence that two sourcing strategies are often considered [11]. From an OSS perspective, sourcing, therefore, regards decisions on if, and what, parts of the internal software that should be based on and/or co-developed as OSS (also referred to as *Open-Sourcing* [5]). This is further highlighted in existing commoditization models (see section 2.3), which argues how commodity parts should be acquired, contributed and sourced in different ways, while internal development should be focused on differenting parts [22, 120]. With this background, we have chosen to base the CAP-model presented in this study on the portfolio purchasing model by Peter Kraljic [104].

Kraljic's model describes how to develop a sourcing strategy for the supply-items (e.g., material and components) required for a product. First, the supply-items are classified according to the *Profit impact* and *Supply risk* dimensions on a scale from low to high. The profit impact concerns the strategic importance of the item, as well as the added value and costs which that it generates for the firm. The supply risk refers to the availability of the item, ease to substitute its suppliers, and how it is controlled. The supply items are then positioned onto a matrix with four quadrants, based on the two dimensions, see Fig. 2. Each quadrant represents

a specific item category with its own distinctive purchasing strategy towards the suppliers [104].

Figure 2: The matrix used in Kraljic's portfolio purchasing model [104], which allows supply-items needed for a product to classified into four item categories based on the two dimensions Business impact and Supply risk.



- **Strategic items:** These are items with high-profit impact and high supply risk. They can usually only be acquired from a single supplier. A common strategy is to form and maintain a strategic partnership with the supplier [25].
- **Leverage items:** These are items with high-profit impact and low supply risk. Can generally be obtained from multiple suppliers at a low switching cost. A common strategy is to exploit buying power within the supplier market [25].
- **Bottleneck items:** These are items with low-profit impact and high supply risk. Suppliers are usually in a dominant position. A common strategy is to accept dependence and strive to reduce negative effects, e.g., through risk analysis and stock-piling [25].
- **Non-critical items:** These are items with low-profit impact and low supply risk. They generally have a low added-value per item. A general strategy is to reduce related costs, such as logistic and administrative [25].

Determining how a material or component should be classified may be done in several ways. Gelderman et al. [68] report how a consensus-seeking method is frequently used by inviting cross-functional competencies and internal stakeholders to discuss how items should be rated in regard to the two dimensions [68]. Other

measurement approaches involve representing each dimension with a specific variable (e.g., supply risk as a number of available suppliers), or using a set of variable and weighting them together. After a set of items have been analyzed and put on the matrix, discussions, and reflections are performed and can potentially lead to a revision of the item categorization [68]. This discussion may concern how the firm should maintain the items' current positions or strive to move certain items between the quadrants.

The model inspired several industries and academics. Among some examples, Caniëls and Gelderman [25] studied the choice of various purchasing strategies and empirically quantified the "relative power" and "total interdependence" aspects among Dutch purchasing professionals. Ulkuniemi et al. [178] looked at purchasing as a market shaping mechanism and identified five types of market shaping actions. Shaya discussed the usage of the Kraljic's portfolio model for optimizing the process of sourcing IT and managing software licenses at Skanska ITN [166]. Gangadharan et al. proposed using Kraljic's portfolio model for mapping SaaS services and sourcing structure [63]. To the best of our knowledge, no study has suggested using Kraljic's model in the context of OSS ecosystems and creation of contribution strategies for software artifacts.

2.5 Strategic Product Planning in OI

A software product strategy defines the product and describes how it will evolve for a longer period of time [62]. It should consider aspects such as the product definition in terms of functional and quality scope, target market, delivery model, positioning and sourcing [4]. Product planning executes product strategy with the help of roadmapping, release planning, and requirements management processes [62]. Hence, decisions regarding if, and what parts of the product should be based on OSS concerns executive management and the software product management (SPM) as they usually oversee the product strategy [122], but also the development organization as they, together with SPM, oversee the product planning and development.

To the best of our knowledge, the current literature offers limited operational support for creating contribution strategies that help synchronize product strategies and product planning with OSS ecosystems. Therefore, we present the CAP model to support software firms in building strategic product planning that looks beyond realizing a set of features in a series of software releases that reflects the overall product strategy and adds the strategic OI aspect with the help of contribution strategies.

[4] http://community.ispma.org/body-of-knowledge/

2.6 Artifacts in Software Engineering

The CAP model presented in this paper offers a tool for firms to decide whether or not a software artifact should be contributed to an OSS ecosystem or not. In this context, a software artifact may refer to a functionality of different abstractions, e.g., bug-fixes, requirements, features, architectural assets or components. These artifacts may be represented and linked together in software artifact repositories [124], often used for gathering, specification and communication of requirements inside a software development organization's requirements management infrastructure [16].

Artifacts may be structured and stored in different ways depending on the context and process used [124]. The resulting artifact structure (also called infrastructure) supports communication between different roles and departments inside an organization, e.g., to which product platform a certain feature belongs, what requirements a certain feature consists of, what test cases that belong to a certain requirement, which release a certain requirement should be implemented in, or what artifacts patches that represent the implementation of a certain requirement. The communication schema should be altered dependent on the firms' needs and processes [58], e.g. to follow-up what requirements are contributed. In this study, we introduce an information meta-model that proposes how a set of repositories may be set up to support the above-mentioned communication and decision-making.

Firms often store software artifacts in a central database and require certain quality criteria in terms of completeness and traceability etc [7]. In contrast, OSS ecosystems constitute an opposite extreme with their usually very informal practices [57]. Here, a requirement may be represented by several artifacts, often complementing each other to give a more complete picture, e.g., as an issue, in a mail thread, and/or as a prototype or a finished implementation. These artifacts are examples of what Scacchi refers to as informalisms [163] and are stored in decentralized repositories (such as issue trackers, mailing lists, and source code repositories respectively).

2.7 Summary

Software engineering has received limited attention in the context of OI, specifically in relation to OSS, which is widespread in practice [141]. Hence, the limited attention that contribution strategies have gotten is not surprising with some exceptions [173, 194]. There is literature explaining general incentives and strategies for how firms should act [40, 79, 188], but neither of the aforementioned or existing models [22, 120] consider aspects specific to OSS, and how firms should synchronize internal product strategy and planning with OSS ecosystem participation [141]. This study aims to address this research gap through a close academia and industry collaboration.

3 Research methodology

In this section, we describe the research design, the process of our study, and our research questions. Further, we motivate the choices of research methods and how these were performed to answer the research questions. Finally, we discuss related validity threats and how these were managed.

3.1 Case Firm

Sony Mobile is a multinational firm with roughly 5,000 employees, developing mobile phones and tablets. The studied branch is focused on developing Android based phones and tablets and has 1600 employees, of which 900 are directly involved in software development. Sony Mobile develops software using agile methodologies and uses software product line management with a database of more than 20,000 features suggested or implemented across all product lines [149].

As reported in earlier work [137], Sony Mobile is a mature OSS player with involvement in several OSS projects. Their existing processes for managing contribution strategies and compliance issues is centrally managed by an internal group referred to as their OSS governance board [137] (cf. OSS Working group [96]). The board has a cross-functional composition as previously suggested with engineers, business managers, and legal experts, and applies the reactive approach as described in section 4.3.

3.2 Research Questions

This study aims to support software-intensive firms involved in OSS ecosystems with integrating their internal product strategy and planning [62] with the decision-process of what software artifacts that they should contribute to the OSS ecosystems, and when, formalized as contribution strategies [194]. Strategic product planning in OI primarily concerns what parts should be revealed (contributed) in an inside-out direction [31] from the firm to the ecosystem. This contribution affects the OSS which in turn is sourced in an outside-in direction [31] from the ecosystem to the firm and is a key enabler in achieving the potential benefits of OI [141]. Earlier research in this area of OI [186], and OSS [141], is sparse and often limited to a management level (e.g., [40, 41, 76, 120]). To occupy this research gap, we aim to design a solution that supports firms in strategic product planning. We pose our first research question (**RQ1**) as:

RQ1: How can contribution strategies be created and structured to support strategic product planning from an OI perspective?

Product planning is a broad practice and usually involves a cross-functional set of internal stakeholders (e.g., legal, marketing, product management, and developers) [103]. This is also the case for strategic product planning and associated

contribution strategies. For a firm with a small development organization, these internal stakeholders may be co-located and efficiently communicate and discuss decisions on a daily basis, but for larger (geographically-distributed) development organizations this may not be possible and cumbersome [43]. A contribution strategy for a certain feature needs to be communicated from the product planning team to the development teams who should implement and contribute accordingly. Conversely, product planning is responsible for monitoring the realization of the approved contribution strategies and what impact they have.

One of the main challenges for market-driven firms is to know what requirements-associated information to obtain, store, manage, and how to enable efficient communication across all stakeholders involved in the crucial decisions that lead to product success [94, 154]. Handling information overload [195] and efficiently connecting the necessary bits and pieces of information is important for strategy realization and follow up analysis. This is particularly important when introducing new concepts that require close collaboration and efficient communication between product management and product development organizations. Thus, RQ2 focuses on the information meta-model that should be integrated into the software artifact repositories used for requirements management and product planning. Our goal is to develop an information meta-model that describes how contributions to OSS ecosystems can be traced to internal product requirements and platforms, and vice versa, and allow for an organizational adoption of contribution strategies for concerned firms. This leads us to pose our second research question (**RQ2**):

RQ2: What software and product planning artifact types and repositories are required and how should they be represented in a meta-model to enable communication and follow-up of contribution strategies in strategic product planning?

By answering these two research questions our goal is to create a practical solution for uncovering further benefits that OI brings [141].

3.3 Research Design and Operation

This study is a design science [81] inspired case study [159]. The work was initiated by problem identification and analysis of its relevance. This was followed by an artifact design process where the artifacts (the CAP model and information meta-model) addressing the research problems (**RQ1 & RQ2**) was created. Finally, the artifacts were validated in the context of the research problem. These steps were performed in close academia-industry collaboration between the researchers and Sony Mobile. We performed data collection and analysis throughout the steps and concluded with reporting of the results (see Fig. 3).

Problem Identification

The objectives of the problem investigation phase in the design process [81] are to further understand the problem context and current practices. To gain greater understanding, we conducted informal consultations with four experts (I1-I4) at Sony Mobile who is involved in the decision-making process of OSS contributions (see Table 1). This allowed us to further refine both **RQ1** and **RQ2** and confirmed their importance and relevance for the industry. Simultaneously, internal processes and policy documentation at Sony Mobile were studied. Next, we received permission to access additional data sources and were able to investigate requirements and contribution repositories. The consultations and investigations confirmed that a suitable solution requires a combination of a technology-based artifact and an organization-based artifact (see guidelines one and two by Hevner [81]). The technology-based artifact (**RQ1**) should allow firms to create contribution strategies for software artifacts and the organizational-based artifact (**RQ2**) should support the organizational adoption and operationalization of the technology-based artifact.

Table 1: Consultation with experts

Expert Id	Years of experience	Role
I1	6 Years	Team Lead
I2	8 Years	Director OSS SW Operations
I3	15 Years	Senior Manager
I4	5 Years	Software Developer

Artifact Design

RQ1 is addressed by designing an artifact that would allow the practitioners to decide whether a software artifact should be contributed to an OSS ecosystem or not. As this is a sourcing issue at the product strategy-level [11, 62, 101], we decided to base the artifact on Kraljic's portfolio purchasing model [104] following the advice and experience of I2 in sourcing. The model consists of a matrix that allows firms to analyze how they source and purchase material and components for their production (see section 2.4).

With this foundation, we iteratively formalized our findings from the consultations with I1-I4 and studies of internal processes and policy documentation. The results of this formalization are the CAP model and the associated meta-model of information required to instantiate the CAP model, supporting strategic product planning in OI. Each item category from the original model [104] has a corresponding type of contribution strategy [194], and instead of supply items, we refer to software artifacts, e.g., features or components. The two dimensions are re-

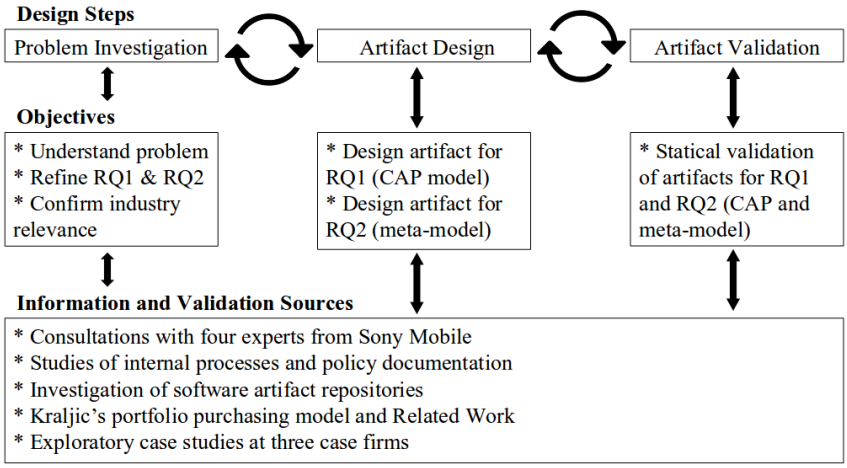


Figure 3: Overview of the research methodology used in this study. The design process was performed iteratively through the three steps involved: problem investigation, artifact design, and artifact valuation [81].

fined to represent *Business impact* and *Control complexity*, inspired by existing commoditization models [22, 120] and literature on OSS ecosystem governance (e.g., [10,41,142]). The measurement process is proposed to employ a consensus-seeking approach [68] with the involvement of cross-functional competencies and internal stakeholders [103]. To help frame the measurement discussion process, questions are defined inspired by literature related to the Kraljic portfolio purchasing model (e.g., [25, 68]), commoditization models [22, 120], software value map (e.g., [9, 97], and OSS ecosystem governance (e.g., [10, 41, 142]). An overlay is created on top of the CAP model to highlight which contribution objective should be the primary driver for the chosen contribution strategy. The objectives represent important value incentives inspired by OI literature [31, 176, 180, 185]. The intention is to help users of the model to fine-tune the contribution strategy for the classified artifact. The CAP model is presented in more detail in section 4.

To address **RQ2** and enable an organizational adoption and operationalization of the CAP-model, we created an information meta-model that facilitates communication and follow-up on software artifacts and their contribution strategies. In the problem investigation phase, it became apparent that the information support should be integrated into the software artifact repositories used for requirements management. The information support would then be able to reach everyone who is involved in the product planning and development. This required us to expand our investigation of Sony Mobile's requirements and contribution repositories, which included a broad set of software artifact repositories that are used in

the product planning of mobile phones. We focused the repository investigation on understanding how contributions could be traced to product requirements and platforms, and vice versa. Through consultation with I1-I4, we selected six relevant repositories: the internal product portfolio, feature repository, feature-based architectural asset repository, patch repository, contribution repository and commit repository (see section 5).

These repositories and their unique artifact IDs (e.g., requirement id, patch id, and contribution id) allowed us to trace the contributions and commits to the architectural assets, product requirements and platforms, via the patches that developers create and commit to internal source code branches. This analysis resulted in the information meta-model presented in Fig. 5. The meta-model creation process was driven by the principles of finding a balance between research rigor and relevance, moving away from extensive mathematical formalizations of the CAP model and focusing on the applicability and generalizability of the model, see guideline five by Hevner [81].

Artifacts Validation

Validation helps confirm that candidate solutions actually address the identified research problems. As we are in an early stage of the research and design process, this study uses static validation [71]. This type of validation uses presentation of candidate solutions to industry practitioners and gathering of feedback that can help to further understand the problem context and refine candidate solutions, in line with the design science process [81]. Dynamic validation [71], which concerns piloting of the candidate solutions in a real-work setting, is a later step in the technology transfer process and is currently under planning at the case firm and is left for future work.

Both the CAP-model and its related information meta-model were validated statically through continuous consultations with experts at Sony Mobile (I1-I4). In these consultations, the models were explained and discussed. Feedback and improvement ideas were collected and used for iterative refinement and improvement. Experts were asked to run the CAP model against examples of features in relation to the four software artifact categories and related contribution strategies that CAP model describes. The examples are presented together with the CAP model and provide further detail and validation of its potential use, see section 4.4. A complete example of how the CAP model and meta-model are used is further presented in section 6. These examples help to evaluate functionality, completeness, and consistency of the CAP model and associated information meta-model. The usability of the information meta-model was further validated by performing traces between the different types of artifacts and their repositories. These traces were presented and used in the static validation of the meta-model. From a design science perspective [81], we employed observational validation through a case study at Sony Mobile where we studied the artifacts (models) in a busi-

ness environment. We also employed descriptive evaluation where we obtained detailed scenarios to demonstrate the utility of the CAP model, see guideline three by Hevner [81].

To improve the external validity of the CAP model, we conducted exploratory case studies at three different case firms (see Section 7). In these case studies, we used static validation [71] where we presented the CAP model to participants from the respective firms and applied it in a simulated setting as part of the interviews. In two of the cases, semi-structured interviews were used with one representative from each firm. In the third case, a workshop setting was used with eight participants from the firm. When collecting feedback from the three case firms, we focused on applicability and usability of the CAP model.

3.4 Ethics and Confidentiality

This study involved analysis of sensitive data from Sony Mobile. The researchers in the study had to maintain the data's integrity and adhere to agreed procedures that data will not be made public. Researchers arranged meetings with experts from Sony Mobile to inform them about the study reporting policies. Data acquired from Sony Mobile is confidential and will not be publicly shared to ensure that the study does not hurt the reputation or business of Sony Mobile. Finally, before submitting the paper for publication, the study was shared with an expert at Sony Mobile who reviewed the manuscript to ensure the validity and transparency of results for the scientific community.

3.5 Validity Threats

This section highlights the validity threats associated with the study. Four types of validity threats [159] are mentioned along with their mitigation strategies.

Internal Validity

Internal validity refers to factors affecting the outcome of the study without the knowledge of the researchers [159].

Researcher bias refers to when the researcher may risk influencing the results in a wanted direction [158]. The proposed CAP model was created with an iterative cooperation between researchers and industry practitioners. Thus, there was a risk of introducing the researcher's bias while working towards the creation of the model. In order to minimize this risk, regular meetings were arranged between researchers and industry experts to ensure the objective understanding and proposed outcomes of the study. Furthermore, researchers and industry practitioners reviewed the paper independently to avoid introducing researcher's bias.

A central part of the CAP model involves estimating the business impact and control complexity. These estimations involve several factors and can have multiple confounding factors that influence them. In this work, we assume that this

threat to internal validity is taken into consideration during the estimation process and therefore is not in the direct focus of the CAP model. Moreover, the CAP model does not prevent additions of new factors that support these estimates.

Triangulation refers to the use of data from multiple sources and also ensuring observer triangulation [158]. In this study, our data analysis involved interpretation of qualitative and quantitative data obtained from Sony Mobile. We applied data triangulation by using Sony Mobile's internal artifacts repositories, documents related to contribution strategies and consultation with relevant experts before proposing the CAP model. There were risks of identifying the wrong data flows and subjective interpretation of interviews. In order to mitigate these risks, concerned multiple experts with different roles and experiences (see Table 1) were consulted at Sony Mobile. We ensured observer triangulation by involving all researchers who authored this manuscript into the data collection and analysis phases.

External Validity

External validity deals with the ability to generalize the study findings to other contexts.

We have focused on analytic generalization rather than statistical generalization [60] by comparing the characteristics of the case to a possible target and presenting case firm characteristics as much as confidentiality concerns allowed. The scope of this study is limited to firms realizing OI with OSS ecosystems. Sony Mobile represents an organization with a focus on software development for embedded devices. However, the practices that are reported and proposed in the study has the potential to be generalized to all firms involved in OSS ecosystems. It should be noted that the case firm can be considered a mature firm in relation to OSS usage for creating product value and realizing product strategies. Also, they recognize the need to invest resources in the ecosystems by contributing back in order to be able to influence and control in accordance with internal needs and incentives. Thus, the application of the proposed CAP model in an other context or in other firms remains part of future work.

The CAP model assumes that firms realize their products based, in part, on OSS code and OSS ecosystem participation. This limits its external generalizability to these firms. At the same time, we believe that the innovation assessment part of the CAP model may be applied to artifacts without OSS elements. In this case, the CAP model provides only partial support as it only helps to estimate the innovativeness of the features (as an innovation benchmark) without setting contribution strategies. Still, this part of the CAP model should work in the same way for both OSS and non-OSS based products. Finally, the classification of software artifacts has a marked business view and a clear business connotation. A threat remains here that important technical aspects (e.g. technical debt, architectural complexity) are overlooked. However, throughout the static validation examples,

we saw limited negative impact on this aspect, especially in a firm experienced in building its product on an OSS platform.

The meta-model was derived from Sony Mobile's software artifact repositories. We believe that the meta-model will fit organizations in similar characteristics. For other cases, we believe that the meta-model can provide inspiration and guidance for how development organizations should implementing the necessary adaptations to existing requirements management infrastructure, or create such, so that contribution strategies for artifacts can be communicated and monitored. We do acknowledge this as a limitation in regards to external validity that we aim to address in future design cycles.

Construct Validity

Construct validity deals with choosing the suitable measures for the concepts under study [159]. Four threats to the construct validity of the study are highlighted below.

First, there was a risk that academic researchers and industry practitioners may use different terms and have different theoretical frames of reference when addressing contribution strategies. Furthermore, the presence of researchers may have biased the experts from Sony Mobile to give information according to researchers' expectations. The selection of a smaller number of experts from Sony Mobile might also contribute to the unbalanced view of the construct.

Second, there was a potential threat to construct validity due to the used innovation assessment criteria based on business impact and control complexity. Both dimensions can be expanded by additional questions (e.g. internal business perspective or innovation and learning perspective [97]) and the CAP model provides this flexibility. One could argue that also technical and architectural aspects should be taken into consideration here. At the same time, the static validation results at Sony Mobile confirm that these aspects have limited importance at least for the studied cases. Still, they should not be overlooked when executing the CAP model in other contexts.

Third, a common theoretical frame of reference is important to avoid misinterpretations between researchers and practitioners [158]. In this study, the Kraljic's portfolio model is used as a reference framework to the CAP model. However, the horizontal and vertical dimensions of Kraljic's portfolio model were changed to control complexity and business impact respectively. Both industry practitioners and academic researchers had a common understanding of Kraljic's portfolio model [104] before discussions in the study. Furthermore, theoretical constructs were validated by involving one of the experts in the writing process from Sony Mobile to ensure consistent understanding.

Fourth, prolonged involvement refers to a long-term relationship or involvement between the researchers and organization [158]. Since there was an involvement of confidential information in the study, it was important to have a mutual

trust between academic researchers and practitioners to be able to constructively present the findings. The adequate level of trust was gained as a result of long past history of collaboration between academic researchers and experts from Sony Mobile.

Reliability

The reliability deals with to what extent the data and the analysis are dependent on the specific researcher and the ability to replicate the study.

Member checking may involve having multiple individuals go through the data, or letting interviewees review a transcript [158]. In this study, the first two authors proposed the meta-model after independent discussions and reviewed by the third author. Furthermore, the model was validated by a team lead, software developer, and senior manager at Sony Mobile, involved in making contributions to OSS communities, were consulted to ensure the correctness of the meta-model and associated data.

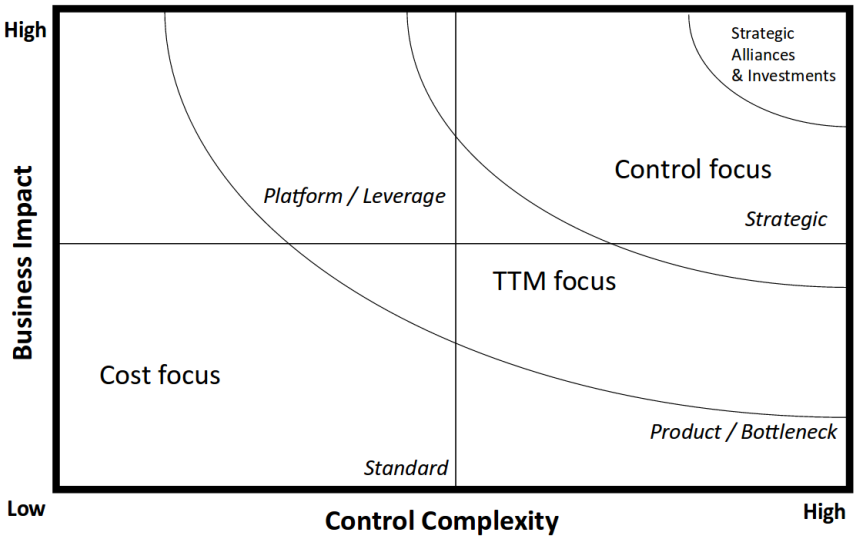
Audit trail regards maintaining traceability between collected data during the study [158]. For this study, the first two researchers kept track of all the mined data from the software artifact repositories as well as the email and informal communication between researchers and Sony Mobile representative. Results were shared with Sony Mobile for any possible misinterpretation or correction of data.

4 The Contribution Acceptance Process (CAP) Model (RQ1)

The CAP model is an adapted version of the portfolio model introduced by Peter Kraljic [104]. Kraljic's model was originally constructed to help firms with creating purchasing strategies towards their suppliers of items needed for their product manufacturing. The CAP model is focused on software artifacts and how these should be sourced and contributed as OSS. The artifacts may be of different abstraction levels, e.g., ranging from specific requirements or issues to sets of requirements as features, frameworks, tools or higher level components.

The model may be used *proactively* or *reactively*. In the former, the model is systematically used on a portfolio or set of artifacts to decide on specific contribution strategies for each artifact, but also to get a general overview and analyze the artifacts relative each other. In the reactive case, the model is used to follow-up on previously classified artifacts, and for individual contribution requests of artifacts from the development organization. We start by describing how the model may be used to classify artifacts and elicit contribution strategies. We then move on and put the model into the context of the two approaches. Lastly, we give examples of artifacts and related contribution strategies.

Figure 4: The Contribution Acceptance Process (CAP) model and its different quadrants that help to determine what contribution strategy to use depending on how a software artifacts are classified in terms of *business impact* and *control complexity*. The overlaying arches marks up four contribution objectives which help to further tailor the contribution strategy (see section 4.1).



4.1 Model Description

The focal point of the CAP model is the matrix presented in Fig. 4. Artifacts are mapped on to the matrix based on how they are valued in regard to the two dimensions *Business impact* and *Control complexity*, located on the vertical and horizontal axis respectively. Business impact refers to how much you profit from the artifact, and control complexity refers to how hard the technology and knowledge behind the artifact is to acquire and control. Both dimensions range from low to high.

Artifact Types and Contribution Strategies

An artifact is categorized into one of the four quadrants, where each quadrant represents a specific artifact type with certain characteristics and contribution strategy. The four types are as follows:

- Strategic artifacts: high business impact and high control complexity.

- Platform/leverage artifacts: high business impact and low control complexity.
- Products/bottlenecks artifacts: low business impact and high control complexity.
- Standard artifacts: low business impact and low control complexity.

Strategic Artifacts: This category includes artifacts that can be internally or externally developed, have a differential value and makes up a competitive edge for the firm. Due to their value and uniqueness, there is a need to maintain a high degree of control over these artifacts. OSS contributions within this category should generally be restricted and made in a controlled manner, ensuring that the differentiation is kept. However, this does not account for possible enablers and/or frameworks, i.e., parts of the artifact that are required for the artifact to work in a given environment. Those have to be actively maintained and contributed. This may require that the artifacts undergo special screening to identify the parts that enable the differentiating parts. In case the artifact is already connected to an existing OSS ecosystem, the firm should strive towards gaining and maintaining a high influence in the ecosystem in regard to the specific artifact and attached functionality. If this is not achievable, e.g., when the contribution terms of an existing ecosystem require contributions to include the differential IP, the option of creating a new and firm-orchestrated OSS ecosystems should be considered. For examples of Strategic artifacts, see section 4.4.

Platform/Leverage Artifacts: These artifacts have a high degree of innovation and positive business impact, but their development does not necessarily need to be controlled by the firm. Examples include technology and market opportunity enablers that have competing alternatives available, ideally with a low switching cost. Generally, everything could be contributed, but with priority given to contributions with the highest potential to reduce time-to-market, i.e., contributions with substance should be prioritized over minor ones, such as error-corrections and maintenance contributions that are purely motivated due to cost reduction. Due to the lower need for control, firms should strive to contribute to existing projects rather than creating new ones, which would require a substantial degree of effort and resources and represent an unnecessary investment. For examples of Platform/Leverage artifacts, see section 4.4.

Products/Bottleneck Artifacts: This category includes artifacts that do not have a high positive business impact by itself but would have a negative effect if not present or provided. For example, functionality firmly required in certain customer-specific solutions but are not made available for the general market. These artifacts are hard to acquire and requires a high degree of control due to the specific requirements. The strategy calls for securing the delivery for the specific customers, while and if possible, sharing the burden of development and maintenance. Generally, everything could be contributed, but with priority given to contributions with the highest potential to reduce time-to-market, or in this case rather

the time-to-customer. But, due to the unique nature of these artifacts, the number of other stakeholders may be limited in existing OSS ecosystems. This may imply that the artifact will be problematic to contribute in a general OSS ecosystem. An option would then be to identify and target specific stakeholders of interest, i.e. of customers and their suppliers, and create a limited project and related OSS ecosystem. For examples of Products/Bottlenecks artifacts, see section 4.4.

Standard Artifacts: This category includes artifacts that may be considered as a commodity to the firm. They do not have a competitive edge if kept internal and has reached a stage in the technology life-cycle where they can create more value externally. They may be externally acquired as easily as internally developed and may, therefore, be considered to have a low level of control complexity. Generally, everything should be contributed, but with priority given to contributions with the highest cost reduction potential. Creating a competing solution to existing ones could lead to unnecessary internal maintenance costs, which has no potential of triggering a positive business impact for a firm. For examples of Standard artifacts, see section 4.4.

Contribution Objectives

Mapping an artifact relative to the four quadrants brings an indication and guideline about its contribution strategy. There are also intrinsic objectives for making contributions that are not fully captured by just accessing the business impact and control complexity in the artifact classification process. These objectives include:

- Cost focus
- Time-to-market (TTM) focus
- Control focus
- Strategic Alliances and Investments

These objectives are closely coupled to the different strategies and are presented as an overlay of the matrix, thus emphasizing the main contribution objective per strategy.

Cost focus: Artifacts with a limited competitive advantage, i.e., they are considered as commodity or enablers for other artifacts, will have a contribution objective mainly focused on *reducing the cost of development and maintenance*. The contribution strategy should focus on minimizing the number of internal patches that need to be applied to each new OSS project release and reusing common solutions available in OSS to fulfill internal requirements, i.e., overall reduce variants and strive for the *standardization* that comes with OSS. As a consequence, internal resources may be shifted towards tasks that have more differentiation value for a firm.

Time-To-Market (TTM) focus: Artifacts that have higher levels of competitive advantage, and/or require a higher amount of control and understanding than commodity artifacts should likely to have the general objective to be advanced to the marketplace as soon as possible, superseding the objective of reducing maintenance costs. These artifacts may also be referred to as *qualifiers*, i.e., artifacts that are essential but still non-differential, and should be contributed as soon and often as possible in order to allow for the own solution to be established as the leading open solution. This will potentially give the advantage of control and barring competing solutions which would otherwise require additional patching or even costly redesigns to one's own product.

Control focus: Artifacts with a high level of competitive advantage and requiring a high level of control are likely to provide differentiation in the marketplace, and should thus not be contributed. Yet, in securing that these artifacts are enabled to operate in an open environment, it is as important to contribute the enabling parts to the OSS ecosystems. If an alternative open solution would become widely adapted out of the firm's control, the firm's competitive edge will likely be diminished and make a costly redesign imperative. Hence, the contribution objective for these artifacts is to take control of the OSS ecosystem with the general strategy to gain and maintain necessary influence in order to better manage conflicting agendas and levy one's own strategy in supporting the artifact.

Strategic Alliances and Investments: These artifacts carry a very large part of product innovation and competitive advantage, and require strict control. Thus, these artifacts should be internally developed, or, if this is not feasible, co-developed using strategic alliances and investments that secure IPR ownership, hence there is generally no objective for making open source contributions.

Adapting Contribution Strategies with Contribution Objectives

Having just a single contribution objective for an artifact is rare except for the extreme cases, e.g., when an artifact is mapped in the far corners of the matrix, such as the bottom left as strictly standard and commodity. More common is to have two or more contribution objectives in play, though one of the objectives would be the leading one. The overlay of contribution objectives on the matrix's different contribution strategies is intended as a guidance for fine-tuning the contribution strategy for individual artifacts when more than one contribution objective is in play. E.g., although two artifacts who are found to have the same overall Platform/Leverage contribution strategy, there might be a degree of difference in the emphasis to be made in the time-to-market objective for an artifact closer to the Strategic area, compared with an artifact closer to the Standard area where considerations on cost of maintenance might overtake as the leading objective.

4.2 Proactive Approach

When proactively using the model, the following step-by-step approach is recommended:

S1 Decision on scope and abstraction level.

S2 Classification and mapping artifacts to the matrix.

(a) Begin with an initial set of artifacts to the matrix.

(b) Synchronize and reiterate mapping.

(c) Map the rest of the artifacts to the matrix.

S3 Reiteration of the artifact mapping.

S4 Documentation and communication of the decisions.

S5 Monitoring and follow-up on the decisions.

Before the model is used, the scope and abstraction level of the analysis needs to be decided (**S1**). The scope may, for example, entails a product, a platform or functional area. Abstraction level concerns the artifacts relative to the scope, e.g., components, features, or requirements. Based on these limitations, the artifacts should be listed, and necessary background information collected, e.g., market intelligence, architectural notes and impact analysis, OSS ecosystem intelligence, and license compliance analysis.

The collected information should then be used as input to an open consensus-seeking discussion forum (**S2**), where relevant internal stakeholders help to classify the artifacts. As in the roadmapping process [103], these stakeholders should bring cross-functional perspective to the decision-making to further explain and argue based on the collected background information, e.g., representatives from marketing, product management, development, and legal.

To facilitate the discussions and help assess the business impact of the artifacts, a set of questions may be used. The joint answers to these questions are given on a Likert scale with values between 1 and 4. The reason for this scale is to force discussion participants to take a clear stand on which side of two quadrants they think an artifact belongs. The questions are as follows (it equals an artifact):

1. How does it impact on the firm's profit and revenue?
2. How does it impact on the customer and end user value?
3. How does it impact on the product differentiation?
4. How does it impact on the access to leading technology/trends?
5. How does it impact if there are difficulties or shortages?

As with the business impact, a set of questions are proposed to help asses the control complexity of the artifact on a scale between 1-4:

1. Do we have knowledge and capacity to absorb the technology?
2. Are there technology availability barriers and IPR constraints?
3. What is the level of innovativeness and novelty?
4. Is there a lack of alternatives?
5. Are there limitations or constraints by the firm?

For an example of how these questions can be used, see section 6. When all questions are answered, the mean values for both dimensions should be calculated. Based on these values, the artifact is then mapped onto the matrix (see Fig. 4), which will put it into one of the four quadrants. The group should then ask themselves if the calculated position agrees with their general belief of where it should be. They should also ask themselves where they want it to be. Further, they should consider what contribution objective(s) that apply, and how this affects the contribution strategy. This process should be allowed to take time and reiteration of the first set of artifacts, as this is necessary for everyone to get accustomed with the process and the classification criteria.

This classification process is not intended to be quantitative and rigorous, but rather qualitative and informal. The process was facilitated through consensus-seeking discussions within a cross-functional group. This approach helps to create guidelines without introducing complexity which may risk introducing negative effects on the usability and applicability of the CAP model. The questions should further be seen as a mean to frame and drive the discussion, during which further questions might come up.

When all artifacts have been classified and mapped onto the matrix, an overall discussion and reflection should be performed (**S3**). When consensus is reached, the decisions should be documented and handed over to product management for communication out to the development organization (**S4**) through required channels supported by the information meta-model, e.g., the requirements management infrastructure (see section 5). The contribution strategies for each artifact should then be monitored and followed-up in a given suitable time frame (e.g., in relation to internal release cycles) (**S5**). This task may be suitable for product or project management with accountability towards the firm's OSS executive.

4.3 Reactive Approach

The CAP model may also be used in a *reactive* mode which is based on Sony Mobile's current practices. This approach is critical in order to continuously follow-up on previously classified artifacts as the classification may change with the artifacts'

technology life-cycle. The approach is also useful for managing individual contribution requests of artifacts from the development organization, e.g. in response when a manager or developer request to contribute a certain artifact, or be allowed to work actively with a specific OSS ecosystem. The CAP model is used in this case by a group of internal stakeholders, similarly to that of the proactive approach. Sony Mobile applies this reactive approach through their OSS governance board (see section 3.1).

When an individual wants to make a contribution, they have to pass through the board. However, to avoid too much bureaucracy and a bottleneck effect, the contribution process varies depending on the size and complexity of the contribution. In the CAP model, the contributions may be characterized in one of three different levels:

- **Trivial contributions** are rather small changes to already existing OSS ecosystems, which enhances the non-significant code quality without adding any new functionality to the system e.g., bug fixes, re-factoring etc.
- **Medium contributions** entails both substantially changed functionality, and completely new functionality e.g., new features, architectural changes etc.
- **Major contributions** are comprised of substantial amounts of code, with significant value in regard to IPR. These contributions are a result of a significant amount of internal development efforts. At Sony Mobile, one example of such a contribution is the Jenkins-Gerrit-trigger plug-in [137].

For trivial contributions, the approval of concerned business manager is sufficient. For medium and major contributions, the business manager has to prepare a case for the Open Source Governance board to verify the legal and IPR aspects of the OSS adoption or contribution. The Open Source Governance board decides after case investigation that include IPR review. Consequently, the board accepts or rejects the original request from the engineers. To further lessen the bureaucracy, Sony Mobile uses frame agreements that can be created for OSS ecosystems that are generally considered as having a non-competitive advantage for Sony Mobile (e.g., development and deployment infrastructure). In these cases, developers are given free hands to contribute what they consider as minor or medium contributions, while major contributions must still go through the board.

4.4 Contribution Strategies with Artifact Examples

In this section, we provide examples in regard to the four artifact types of the CAP model, which we elicited from consultations with experts from Sony Mobile.

Strategic Artifacts:

Example 1 - Gaming, Audio, Video, and Camera: A typical example of a contributable enabler is multimedia frameworks which are needed for services such

as music, gaming, and videos. The frameworks themselves are not of a strategic value, but they are essential for driving the Sony brand proposition since they are needed in order to provide the full experience of strategic media and content services provided by Sony. Such artifacts may also be referred to as Qualifiers, as they are essential, yet not strategic by themselves.

An example of such a multimedia framework that Sony Mobile uses is AndroidâĂŹs Stagefright[5]. It is for example used for managing movies captured by the camera. The framework itself could be contributed into, but not specific camera features such as smile recognition as these are considered as differentiating towards the competition, hence have a high business impact and control complexity for Sony Mobile. In short, camera effects can not be contributed, but all enablers of such effects should be, thus Sony Mobile contributes to the frameworks to steer and open up a platform for strategic assets, e.g., an extended camera experience on their mobile phones. A further example of a framework that has been made open by Sony, but in the context of gaming, is the Authoring Tools Framework[6] for the PlayStation 4.

Platform/Leverage Artifacts

Example 1 - Digital Living Network Alliance: Digital Living Network Alliance (DLNA) (originally named Digital Home Working Group) was founded by a group of consumer electronics firms, with Sony and Intel in leading roles, in June 2003. DLNA promotes a set of interoperability guidelines for sharing and streaming digital media among multimedia devices.

As support for DNLA was eventually included in Android, creating a proprietary in-house solution would not have been wise given that the OSS solution already was offered. Instead, Sony Mobile chose to support the Android DNLA solution with targeted but limited contributions. This is a typical example of leveraging functionality that a firm does not create, own, or control, but that is good to have. Hence, Sony Mobile did not need to commit extra resources to secure the interoperability of an own solution. Instead, those extra resources could be used for making the overall offering better, e.g., the seamless streaming of media between Android devices and other DNLA compliant device, for instance, a PlayStation console, and in that way promote DNLA across SonyâĂŹs all device offerings.

Example 2 - Mozilla Firefox: The most significant web browsers during the 1990s were proprietary products. For instance, Netscape was only free for individuals, business users had to pay for the license. In 1995, Microsoft stepped into browser market due to the competitive threat from Netscape browser. Microsoft decided to drive the price of web browsers market by bundling its competitive browsers for free with the Windows operating system. In order to save the market share, Netscape open sourced the code to its web browsers in 1998 which resulted

[5] https://source.android.com/devices/media/
[6] https://github.com/SonyWWS/ATF

in the creation of the Mozilla organization. The current browser known as Firefox is the main offspring from that time. By making their browsers open source, Netscape was able to compete against Microsoft's web browsers by commoditizing the platform and enabling for other services and products.

Products/Bottleneck Artifacts

Example 1 - Symbian network operators requirements: In the ecosystem surrounding the Symbian operating system, network operators were considered one of the key stakeholders. Network operators ran the telephone networks to which Symbian smart-phones would be connected. Handset manufactures are dependent on the operators for distribution of more than 90% of the mobile phone handsets, and they were highly fragmented, with over 500 networks in 200 countries. Consequently, operators can impose requirements upon handset manufactures in key areas such as pre-loaded software and security. These requirements can carry the potential to one of those components that do not contribute in terms of a business value but would make a negative impact on firm's business if missing, e.g., by a product not being ranged.

Example 2 - DoCoMo mobile phone operator: DoCoMo, an operator on the Japanese market, had the requirement that the DRM protection in their provided handsets uses Microsoft's PlayReady DRM mechanism. This requirement applied to all handset manufacturers, including Sony Mobile's competitors. Sony Mobile, who had an internally developed PlayReady plug-in, proposed that they could contribute it as OSS and create an ecosystem around it and also because it already contributed the DRM framework. DoCoMo accepted, which allowed Sony Mobile and its competitors to share maintenance and development of upcoming requirements from DoCoMo. In summary, Sony Mobile solved a potential bottleneck requirement which has no business value for them by making it OSS and shared the development cost with all its competitors while still satisfying the operator.

Standard Artifacts

Example 1 - WiFi-connect[7]: This OSS checks whether a device is connected to a Wi-Fi. If not, it tries to join the favorite network, and if this fails, it opens an Access Point to which you can connect using a laptop or mobile phone and input new Wi-Fi credentials.

Example 2 - Universal Image Loader[8]: Universal Image Loader is built to provide a flexible, powerful and highly customizable instrument for image loading, caching and displaying. It provides a lot of configuration options and good control over the image loading and caching process.

[7] https://github.com/resin-io/resin-wifi-connect
[8] https://github.com/nostra13/Android-Universal-Image-Loader

Both examples are considered standard artifacts because they can be considered as a commodity, accessible for competition and do not add any value to customers in the sense that they would not be willing to pay extra for them.

5 Operationalization of the CAP model (RQ2)

Putting contribution strategies into practice requires appropriate processes and information support to know which artifacts, or what parts of them that should be contributed. Furthermore, to follow up the contribution strategy execution and make necessary adaptations as the market changes, there needs to be a possibility to see what has been contributed, where, and when. In this section, we address research question **RQ2** and propose an information meta-model which can be used to record and communicate the operationalization of the CAP model, e.g., by integrating it into the requirements management and product management information infrastructure.

The meta-model was created through an investigation of Sony Mobile's software and product management artifact repositories used in product planning and product development. During this investigation, we focused on how the contributions could be traced to product requirements and platforms, and vice versa. Through consultation with I1-4, the investigation resulted in the selection of six repositories, see Fig. 5:

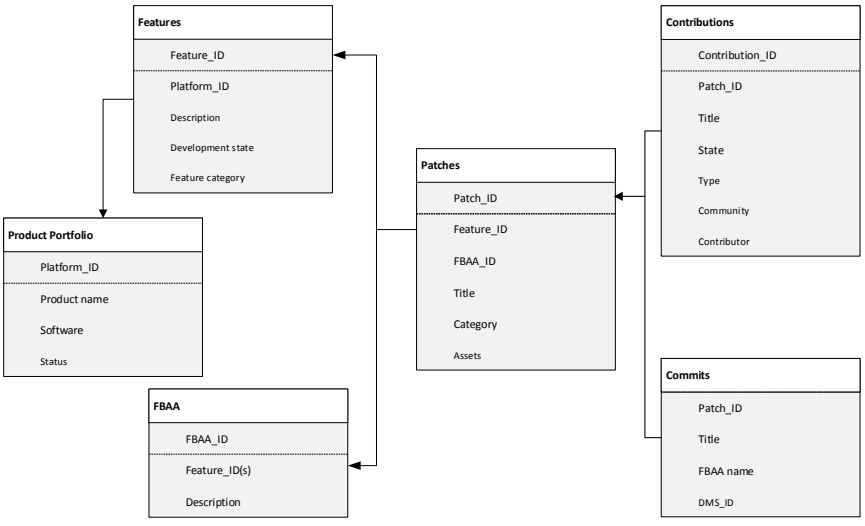


Figure 5: Software artifact repositories necessary to communicate and follow-up on contribution strategies decided with the CAP model.

- Product Portfolio repository
- Features repository
- Feature-Based Architecture Assets repository
- Patch repository
- Contribution repository
- Commit repository

These repositories and their unique artifact ids (e.g., requirement id, patch id, and contribution id) allowed us to trace the contributions and commits to their architectural assets, product features, and platforms, via the patches that developers create and commits to internal source code branches. Table 2 presents the repositories including their attributes.

The product portfolio repository is used to support Sony Mobile's software platform strategy, where one platform is reused across multiple phones. The repository stores the different configurations between platforms, hardware and other major components along with market and customer related information. **The feature repository** stores information about each feature, which can be assigned to and updated by different roles as the feature passes through the firm's product development process. Information saved includes documentation of the feature description and justification, decision process, architectural notes, impact analysis, involved parties, and current implementation state. The *contribution strategy* attribute is used to communicate the decisions from the CAP model usage, on whether the feature should be contributed or not.

Feature-Based Architectural Asset (FBAA) repository (FBAAs) groups features together that make up common functionality that can be found in multiple products, e.g. features connected to power functionality may be grouped together in its own FBAA and revised with new versions as the underlying features evolve along with new products. Products are defined by composing different FBAAs which can be considered as a form of configuration management.

Even though Sony Mobile uses Android as an underlying platform, customization and new development are needed in order to meet customers' expectations. These adaptations are stored as patch artifacts in the **patch repository**. The patch artifacts contain information about the technical implementation and serve as an abstraction layer for the code commits which are stored in a separate **commit repository**. Each patch artifact can be traced to both FBAAs and features.

The patches that are contributed back to the OSS ecosystems have associated contribution artifacts stored in the **contribution repository**. These artifacts store information such as the type of contribution and complexity, responsible manager and contributor, and concerned OSS ecosystem. Each contribution artifact can be traced to its related patch artifact.

Table 2: Description of selected attributes from the software artifact repositories mentioned in Fig. 5

Repository Name	Attributes	Description
Products	Platform ID	A unique ID for platform name
	Product name	Product name with the platform.
	Software	Related software description, e.g., Android, OSE, Epice, Kept etc.
	Status	Current standing of the platform, e.g., expired, announced etc.
Features	Feature ID	A unique Id for a feature, which refers to features.
	Platform ID	ID associated with the specific platform e.g. android, core etc.
	Description	Details of the feature.
	Development state	Refers to the current status a feature's implementation, e.g., started, executed.
	Feature category	Refers to the type of feature, e.g., new functionality, bug fix, extension etc.
	Contribution Strategy	Refers to whether the requirement is contributable or not.
FBAA	FBAA ID	A unique Id for each Feature Based Architecture Asset (FBAA).
	FP IDs	A combination of FP IDs associated with the FBAA.
	Description	Details of a FBAA.
Patches	Patch ID	A unique id for each patch.
	FP ID	A unique ID from the FP repository.
	FBAA ID	A unique ID from the FBAA repository.
	Title	A description of a patch.
	Category	Importance of a patch, e.g., market critical, development critical, stability, ecosystem critical etc.
	Assets	Refers to the type of a patch, e.g., bug fix, extension, operator requirement, platform related, generic etc.
Contributions	Contribution ID	A unique ID for each contribution.
	Patch ID	A unique ID from the patches repositories.
	Title	A description of a contribution.
	State	Refers the current state of the patch, e.g., ecosystem merged, already fixed, CEO rejected, legal reject, ecosystem review etc.
	Type	Refers to criticality of a contribution, e.g., trivial, non-trivial, bug fix etc.
	ecosystem	Refers to the ecosystem in which the contribution will be made, e.g., Google, Firefox etc.
	Contributors	Refers the contributor information.
Commits	Patch ID	A unique Id from the patch repository.
	Title	A detailed description of a commit.
	FBAA name	Commits associated with the FBAA.

With this set-up of repositories and their respective artifacts, Sony Mobile can gather information necessary to follow up on what functionality is given back to OSS ecosystems. Moreover, Sony Mobile can also measure how much resources that are invested in the work surrounding the implementation and contribution. Hence, this set-up makes up a critical part in both the structuring and execution of the CAP model.

This meta-model was created in the context of Sony Mobile's development organization. Hence, it is adapted to fit Sony Mobile' software product line strategy with platforms from which they draw their different products from. The architectural assets (FBAAs) play a key part in this configuration management. As highlighted in section 3.5, we believe that the meta-model will fit organizations in similar characteristics, and for other cases provide inspiration and guidance. This is something that we aim to explore and validate beyond Sony Mobile in future design cycles.

6 Combining the CAP Model and the Information Meta-model

In this section, we provide an example of how the CAP model may be used to classify an artifact, and combine this with the information meta-model to support communication and follow-up of the artifact and its decided contribution strategy. The example is *fictive*[9] and was derived together with one of the experts (I2) from Sony Mobile with the intention to demonstrate the reasoning behind the artifact classification. Following the proactive process defined in section 4.2, we begin by discussing scope and abstraction level.

For Sony Mobile, FBAAs offer a suitable abstraction level to determine whether certain functionality (e.g., a media player or power saving functionality) can be contributed or not. If the artifact is too fine-grained it may be hard to quantify its business impact and control complexity. In these cases, features included in a certain FBAA would inherit the decision of whether it can be contributed or not. Regarding the scope, we look at FBAAs related to the telephony part of a certain platform-range. The FBAA that we classify regards the support for Voice over Long-Term Evolution (VoLTE), which is a standard for voice service in the LTE mobile radio system [151]. Note that this classification is performed when VoLTE was relatively new to the market in 2015.

VoLTE is classified in regard to its business impact and control complexity. The questions defined in section 4.2 were used. Under each question, we provide a quote from I2 about how (s)he reasons, and the score which can be in the range of 1-4. We start by addressing the business impact:

[9] Due to confidentiality reasons, we have to select this example.

1. How does it impact on the firm's profit and revenue?
   "VoLTE is hot and an enabler for services and the European operators are very eager to get this included. This directly affects the firm's ability to range its products at the operators. So very important. Is it super important? The consumers will not understand the difference of it, they will get it either way." - **Score: 3**.
2. How does it impact on the customer and end user value?
   "The consumers themselves may not know about VoLTE, but they will appreciate that the sound is better and clearer because other coding standards may be used." - **Score: 3**.
3. How does it impact on the product differentiation?
   "VoLTE has a positive effect. Some product vendors will have VoLTE enabled and some not. So there is a differentiation which is positive. Does this have a decisive effect concerning differentiation? Is it something that the consumers will interpret as something that is very important? No." - **Score: 3**.
4. How does it impact on the access to leading technology/trends?
   "VoLTE is very hot and is definitely a leading technology." - **Score: 3**.
5. How does it impact if there are difficulties or shortages?
   "If we cannot deliver VoLTE to our customers, how will that affect them? It will not be interpreted as positive, and will not pass us by. But they will not be fanatic about it." - **Score: 2**.

This gives us a mean score of 2,8. We repeat the same process for control complexity:

1. Do we have knowledge and capacity to absorb the technology?
   "Yes, we have. We are not world experts but we do have good knowledge about it." - **Score: 3**.
2. Are there technology availability barriers and IPR constraints?
   "Yes, there were some, but not devastating. There are patents so it is not straight forward." - **Score: 2**.
3. What is the level of innovativeness and novelty?
   "It is not something fantastic but good." - **Score: 3**.
4. Is there a lack of alternatives?
   "Yes, there are not that many who have development on it so there are quite a few options. So we implemented a stack ourselves." - **Score: 3**.
5. Are there limitations or constraints by the firm?
   "No, there are none. There is not a demand that we should have or need to have control over." - **Score: 1**.

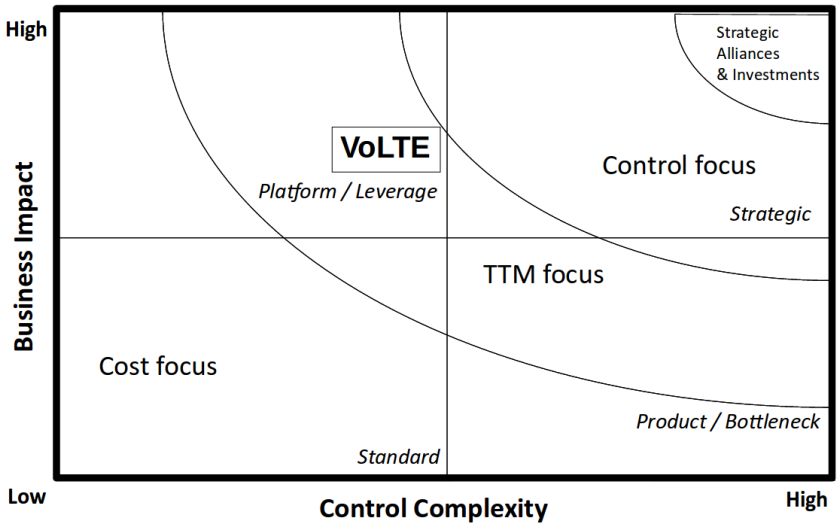


Figure 6: The CAP model and the example of VoLTE which is classified in regard to its business impact and control complexity.

This gives us a mean score of 2,4. This places VoLTE in the bottom between of the upper two quadrants; the strategic and platform/leverage artifact quadrants. I2 elaborates on the strategy chosen:

"VoLTE is an opportunity for us. We should invest in this technology, but we do not have to develop our own solution. Rather, we should take what is available externally. We should do active contributions, not just to get rid of maintenance, but also to push the technology forward with a time-to-market as our main contribution objective. It does not matter if it is open source. This is not rocket science that only we know about. We should have an open attitude towards VoLTE and support it as OSS and invest in it."

After reiterations and discussions, the decisions should be documented and communicated to the development organization. In Sony Mobile's case, the information meta-model is already integrated into requirements and product management infrastructure. Thus, these decisions would be added to the contribution strategy attribute of the feature artifacts which belong to the VoLTE FBAA artifact. To monitor and follow-up on the contribution strategy execution for VoLTE, product management can trace patch artifacts connected to the VoLTE feature artifacts, and see which of these that have contribution artifacts connected to them.

7 Case studies

To perform a first validation of the CAP model outside Sony Mobile, we have conducted three exploratory case studies where we applied the CAP model and investigated its applicability and usability. Further and more extensive application and validation are planned for future design cycles. Below we present the results from this validation per case firm, which due to confidentiality reasons are made anonymous and referred to as firm A-C. For each firm, we present general characteristics, and how we conducted the case study. We then give a brief overview of their overall contribution strategy, followed a summary of the application of the CAP model, and an evaluation of the model in terms of its usability. For an overview, see table 3.

7.1 Case Firm A

Firm A operates in the agriculture business. The main product of the firm is software designed to improve the efficiency of global grain marketing. The software offers a communication platform between the growers and buyers combined with real-time market intelligence. The main benefit is an enhanced ability to quickly respond to domestic and global market demands. We interviewed the CTO of the firm who has over 25 years of experience in the IT sector and was involved in 10 start-ups and many projects. The CAP model was used to analyze the current product the firm is offering.

Table 3: Overview of the three case firms in regard to their domain, use of OSS, scope and abstraction analyzed with the CAP, and the setting in which the model was applied.

	Description	**Use of OSS**	**Scope & Abstraction**	**Setting**
Firm A	Small-sized firm building a platform product for the agricultural domain.	OSS components in platform.	Features in platform product.	Interview with CTO.
Firm B	Small-sized firm building mobile games for mobile platforms.	OSS components in game products.	Features in a specific game.	Interview with Founder.
Firm C	Large-sized firm in the telecommunication domain.	OSS in service infrastructure.	Internal infrastructure project	Workshop with 8 cross-functional participants.

Overall Contribution Strategy

The firm makes extensive use of OSS code as long as it is not released under the GPL version 3 license. The firm keeps its own copy of the source code and often contributes bug fixes or other small changes, however without following up if they are integrated into the common code base. Decisions if to adapt the OSS ecosystem's version of the code are made on regular basis upon analysis.

The firm has currently a static code policy that is based on the following reasoning. If the existing code works at the time, the firm does not care if it evolves and does not check if never versions are available. If there are changes, the firm checks first if the suggested improvements are beneficial before any new version is considered and integrated.

Maintenance cost reduction is important for the firm, however not for the price of loosing competitive advantage. Thus, any functionality that has a differentiating potential is kept proprietary for about 6-9 months to check the market response and profitability. After this time, the firm analyzes if cost reduction is substantial before deciding to contribute the code or not. Estimating the current or future value of an asset is challenging, mainly because of rapid market changes and high market uncertainty. An example here is inventory management module that the firm's product has. This module (feature) turned out to be a strategic asset 12 months after developing it. So what may seem to be a rational decision from the development/technology perspective can be overwhelmed by market forces or conditions. Moreover, it may take a substantial amount of time before an intellectual prop-

erty asset reveals its true value in the market place due to delays in the technology adoption curve. Therefore, cautious evaluation of the business and revenue values are necessary. If the technology adoption is slow, it is much more challenging and harder to see if and when to contribute.

Regarding the contribution strategy, the firm has the following rules:

- high profit and critical to maintain control features are never shared with the OSS ecosystem as these build the firm's value in the eyes of the shareholders
- high profit and not critical to maintain control features - some resources are dedicated to investigate and see the potential of growing from low profit to high profit before a decision to contribute is made
- low profit and critical to maintain control features - the firm can release these features after commodity analysis.
- low profit and not critical to maintain control features - the firm contributes these features as quickly as possible.

The firm is small and in a growing phase with limited resources that can be dedicated to working with the OSS communities. The conclusion here is that OSS ecosystem engagement can be very valuable for large enterprises, in a resource constrained enterprise it is pretty risky policy.

Application of the CAP Model

Together with the firm's CTO, we have analyzed the current product with the help of the CAP model. The mapping of the product's features on the CAP model brings into focus the questions regarding: 1) where the differentiating value is, 2) what is the nature of the market the firm is operating in and 3) how much value the potential customers can absorb. This resulted in the following categorization:

- **Standard artifacts** - Covers about 20% of all features. The CTO adds that not only OSS software is considered here but also binary modules.
- **Product/Bottleneck artifacts** - Covers about 20% of all features. These are mostly purchased or obtained from OSS communities to a lower time-to-market. An interesting aspect here is the usage of binaries that further reduces time-to-market as the integration time is lower compared to OSS modules that often require some scripting and integration efforts.
- **Strategic artifacts** - Covers only about 5% of all features. The main reason is that the firm is afraid someone will standardize or control something in that part (interfaces) and destroy the shareholders' value.
- **Platform/Leverage artifacts** - Covers about 55% of all features because complexity is low and the firm has high control in case the firm becomes dominant in the market (they are currently not dominant).

According to the CTO, a firm can be a "big winner" in immature markets that usually lack standards. Having a high portion of features in the Strategic artifact corner indicate operating in an established market where alliances need to be made do deliver substantial value.

Usability of the CAP Model

The CTO indicated that the CAP model can be used by both executives and operational management. The primary stakeholder remains everyone who is responsible for product strategies. However, the executives will focus mostly on the strategy and if it reflects the direction given by the Board of Directors and main shareholders. In that regard, the percentage mapping of the features on the CAP model is considered useful as it shows where in those four quadrants (see Fig. 4) a firm's product is, but also where it should be. When applied, there should be a cross-functional group as earlier suggested (see section 4). The CTO agrees that a consensus-seeking approach should be used where opinions are first expressed independently, shared and then discussed until the group converges. This shows potential risks and additional uncovered aspects.

When classifying artifacts in terms of business impact and control complexity, the CTO indicated that high-medium-low is sufficient in terms of scale. When several people perform the estimations, the results can show the density of each level for each aspect. The levels should be augmented with comments regarding additional risks or other important aspects. A scale of -1, 0 and 1 was also considered as suitable.

The used frequency of the CAP model is estimated to be every major revision cycle when new features are added to the product. The complete analysis based on the CAP model should be performed when, e.g., entering the new market place or moving to more stable places in the market place.

Our respondent believes that the CAP model usage delivers greater confidence that the firm is not deviating from the strategic direction and helps to identify the opportunities in the area in other quadrants. The usefulness was estimated as high and could be improved with more guidelines on how to interpret the mapping results. At the same time, it appears that larger organizations can benefit more from the CAP model application. The main problem for smaller firms with reaching high utility of the CAP model would be to have the resources to do regular analysis and the experience to provide valuable opinions. Experience in working with OSS and knowledge of the main driving forces for commoditization is considered essential.

7.2 Case Firm B

Firm B develops mobile games for the Android and iOS platforms. The market place that the firm operates in is rather disordered and characterized by several

players who use the same game engine that has a very active ecosystem[10] around it. A substantial part of the product is available for free with little integration effort. Reusing platforms and frameworks with large user base is an important survival aspect, regardless if they are OSS or not since acquisition costs are marginal. Entry barriers are negligible which implies that the commercial success is often a "hit and miss". In many aspects, the environment resembles an inverted OSS ecosystem where a given tool from a given provider or a given module is available with the source. Where a given tool or module from a given provider is available, often with source, at little or no charge. As a result, significant elements of the games are, essentially, commodities and product differentiation principally occurs within the media assets and the gameplay experience. The tool provider[11] is open sourcing back to the ecosystem and can gain those inverted benefits. The customers are helping the provider to improve the quality of the offering. The studied firm only report bugs to these ecosystems and never considers any active contributions or extensions.

The mobile game users expect to play the game for free and perceive them as commodities. This impacts profitability and ability to be commercially viable. If the game is successful there are many opportunities to disturb the market place, e.g. a competitor copies the first 5 levels of the game and offers a similar copy to the market. About 80% of the revenue is generated in the first five days after the game is released since the immediate customer behavior defines if the asset is worth something or not.

Overall Contribution Strategy

Since profitability decreases rapidly after product launch, firm B wants to directly minimize maintenance costs. This implies contributing the code base or using commodity parts as much as possible. Contribution strategy associated decisions need to be made rapidly based on the revenue trends and results. The odds of having long term playability for games other than adventure are very low. So for each release, the firm can receive a spike in the income and profitability and needs to carefully plan how to utilize this income. Time to market remains the main success factor in this market segment.

Analyzing this market segment with the help of the CAP model brings forward how extreme the risk levels are in the mobile games business. CAP works well here as a risk assessment tool that should be applied to investments. In this market place, the quadrants of the CAP model can be merged and discussed together. The main analysis should be along the Y-axis and the discussion should be profit driven since the firm does not have any control over the platform, but controls the player experience.

Regarding the contribution strategy, the firm has the following rules:

[10] https://unity3d.com/
[11] https://unity3d.com/

- high profit and critical to maintain control features - these features are considered as key differentiators but in this context there are very low barriers to copying by fast followers that clone the features. So keeping the features proprietary does not eliminate the risk of "fast clones".
- low profit and not critical to maintain control features - firm B obtains these features from 3rd party suppliers.
- low profit and not critical to maintain control features - firm B tried to obtain the components from 3rd parties and if it is not possible the software architecture is changed to eliminate criticality.
- high profit and not critical to maintain control features - there are no features with this characteristics according to firm B.

Application of the CAP Model

We mapped the product features to the CAP model grid. The results are: 0% of the features in the low left quadrant (**Standard artifacts**), 15% in low right quadrant (**Product/Bottleneck artifacts**), 80% in upper left quadrant (**Platform/Leverage artifacts**) and 5% in top right (**Strategic artifacts**). Because firm B works cross platform they are dependent on the platform provider and obtain other modules from the ecosystem, e.g. the 2d elements and the networking elements. Firm B hopes that remaining focused on the upper left corner is sufficient to get some customers. The firm is "at the mercy of" the other firms dominating the top right corner. CAP helps to points out here that the vast bulk of the technology that enables the experience is already a commodity and freely available so the only differentiating side is the game experience, but this is substantial investment in media, marketing, UI, graphics, and art-work.

Usability of the CAP Model

The CAP model helps to raise attention that the market is very competitive. The commodity price is very low, differentiation is difficult and acquisition costs are marginal. For firm B, it means that it is cheaper to pay someone else for development than to participate in OSS migration and integration. The main benefit from CAP application remains the conclusion that in mobile game development the focus needs to be on business impact. It is important to perform extensive analysis on the Y-axis for checking if a future game is commercially viable before analyzing the complexity dimension.

The CAP model, in this case, can be used once and the clear conclusion for the firm is that it should change its market focus. The model clearly points out that if a firm is relatively new to mobile game development there is little profitability in this market unless you have 20-30 million dollars to invest in marketing and other

actions to sustain long terms revenues. Our respondent believes that every new game concept can be and should be evaluated with the help of the CAP model.

Our respondent believes that the questions in the CAP model should be answered with the high, medium and low scale during a consensus-driven discussion. Since most of the discussion in on the Y-axis, the simple 3-point scale was considered sufficient. Our respondent also pointed out that the CAP model could potentially be extended to include hedonic qualities since a firm sells experience rather than software applications. Investing in a high complex game is very risky so firms in this domain tend to stay away from high complexity endeavors that are risky.

7.3 Case Firm C

Firm C operates in the telecommunication domain and extensively uses OSS to deliver software products and services. We applied the CAP model on one of the internal software infrastructure projects with the objective to support the decision process in regard to whether the project should be released as OSS. CAP was therefore used on a project level, instead of a set of features. We invited 8 participants from various functions at the firm (open source strategy, community management, legal, product management, and development) into a workshop session where the CAP model was discussed and applied.

Overall Contribution Strategy

Decisions on what projects that are released as OSS and what may be contributed to existing OSS projects are made by the OSS governance board, similar to that of Sony Mobile (see section 4.3). The board is cross-functional and includes the representatives from OSS strategy, legal, technology and software development.

Contribution requests are submitted to the OSS governance board from the engineering teams and usually concern projects related to the development toolchain or the infrastructure technology stack. The requests are usually accepted given that no security threats are visible or potential patents can be disclosed. In addition, the board analyses the potential for creating an OSS ecosystem around the project to be released.

Application of the CAP Model

The studied project was first discussed in terms of its background and functionality in order to synchronize the knowledge level among the workshop participants. This was followed by a discussion of the project's business impact. The questions outlined in Section 4.2 were used for framing the discussion, but instead of using the Likert scale of 1-4, the workshop participants opted for an open consensus-seeking discussion from start.

The workshop participants agreed that the project has a high impact in terms profit and revenue, as it increases operational efficiency, decreases the license-costs, and increases security. As it is an internal infrastructure project used to deliver software products and services, it has limited impact on the customers and end-users. The technology is not seen as differentiating towards competitors but does enable easier access to new technology-standards that may have a substantial impact on the business. The firm's engineering department has managed to perform the daily operations and deliver the firm's services without the use of the project, why it would not devastate business if it was no longer available. However, it does offer clear advantages which would cause a negative impact if it the availability was reduced or removed.

In regard to control complexity, it was concluded that the firm has the competence needed to continue developing the project. Further, the project did not include any IP and patents from the firm's defensive patent portfolio. The underlying knowledge and technology can be considered as commodity. However, there is a lack of alternates as only two could be identified, both with shortcomings. Internally of the firm, there is a defined need for the project, and that influence on its development is needed. There is, however, no demand that the firm should maintain absolute control, or act as an orchestrator for the project.

The workshop participants classified the project as a strategic artifact due to the high business impact, as well as a relative need for control and lack of alternatives. Due to the latter reasons, the project should be released as a new OSS ecosystem as soon as possible in order to maintain the first-mover-advantage and avoid having to adapt to competing solutions. Hence, the main contribution objective should be to reduce time-to-market. The participants stated that the goal would be to push the project towards commodity, where the main objective would be to share the maintenance efforts with the ecosystem and refocus resources on more value-creating activities.

Usability of the CAP Model

The workshop participants found that the CAP model provided a useful lens through which their OSS governance board could look at contribution requests and strategically plan decisions. One participant expressed that the CAP model offers a blue-print to identify what projects that are more important to the firm, and align contribution decisions with internal business and product strategies by explicitly considering the dimensions of business impact and control complexity.

The workshop-participants preferred the open consensus-seeking discussions as a mean to determine the business impact and control complexity, and based on this classify the artifact to the most relevant artifact type and contribution strategy. The chosen strategy and aligning contribution objective could then be used to add further depth and understanding to the discussion, which helped the group to arrive at a common decision and final contribution strategy for the reviewed project.

The questions defined in section 4.2 were found useful to frame the discussions. Participants expressed that these could be further customized to a firm, but that this should be an iterative process as the OSS governance board applies the CAP model when reviewing new projects. The participants further expressed that some questions are more relevant to discuss for certain projects than others, but they provide a checklist to walk through when reviewing a project.

8 Discussion

In this section, we discuss the applicability and usability of the CAP model. We discuss the findings from the case studies how the CAP model should be improved or adapted to fit other contexts.

8.1 Applicability and Usability of the CAP Model

The three cases presented in section 7 bring supporting evidence that the CAP model can be applied on: a set of features, a product or on a complete project. The model has proven to bring useful insights in analyzing a set of features in a product with the indication that larger organizations can benefit more from the CAP application than small organizations. In case B, the application of CAP provided valuable insights regarding the nature of the market and the risks associated with making substantial investment in this market. In case C, the application of the CAP model provide a lens though which the OSS governance board can screen current projects and decide upon their contribution or OSS release strategies.

CAP was found useful as decision-support for individuals, executives and managers. However, as highlighted by respondents from firms A, B and C, CAP is best suited for a cross-functional group where consensus-seeking discussions can be used to bring further facets to the discussions and better answer the many questions that needs to be addressed. As for Sony Mobile and case firm C, a suitable forum for large-sized firms would be the OSS governance boards or OSS program offices.

The questions suggested in section 4.2 were found useful, but it was highlighted that these may need to be tailored and extended as CAP is applied to new projects and features. When answering the questions and determining the dimensions of business impact and control complexity, the cases further showed that on scale does not fit all. Case firm A and B suggested a high-medium-low scale, while case firm C preferred to use the consensus-seeking discussion with out the help of a scale. These facts highlight that certain adaptations are needed for the CAP model to maintain its usability and applicability in different settings. It also highlights that the decision process should not be "over-engineered". Our results suggest that complexity needs to be balanced in order to maintain usability for the practitioners while still keeping the applicability on different types of artifacts and

settings. How to adapt this balancing act and tailor the CAP model to different settings is a topic for future design cycles and case evaluations.

8.2 Influence Needed to Control

The Kraljic's portfolio model was originally used to help firms to procure or source supply-items for their product manufacturing [104]. One of the model's two decision factors is *supply risk*. To secure access to critical resources, a certain level of control is needed, e.g., having an influence on the suppliers to control the quality and future development of the supply-items. For OSS ecosystems, this translates into software engineering process control, for example in terms of how requirements and features are specified, prioritized and implemented, with the goal to have them aligned with the firm's internal product strategy.

Software artifacts with a high control complexity (e.g., the media frameworks for Sony Mobile, see section 4.4) may require special ownership control and a high level of influence in the concerned OSS ecosystems may be warranted to be able to contribute them. In cases where a firm does not posses the necessary influence, nor wish to invest the contributions and increased OSS activity [41] which may be required, an alternative strategy is to share the artifact with a smaller set of actors with similar agendas, which could include direct competitors [188]. This strategy is still in-line with the meritocracy principle as it increases the potential ecosystem influence via contributions [41]. Sharing artifacts with a limited number of ecosystem actors leaves some degree of control and lowers the maintenance cost via shared ownership [176, 180]. Further, time-to-market for all actors that received the new artifacts is substantially shortened.

For artifacts with less complexity control, e.g., those concerning requirements shared between a majority of the actors in the OSS ecosystem, the need for control may not be as high, e.g., the DLNA project or Linux commodity parts, see sections 4.4 and 4.4. In these cases, it is therefore not motivated to limit control to a smaller set of actors which may require extra effort compared to contributing it to all ecosystem actors. An alternative implementation may already be present or suggested which conflicts with the focal firm's solution. Hence, these types of contributions require careful and long term planning where the influence in the ecosystem needs to be leveraged. In case of firm B, complexity is controlled by the framework provider.

For both critical or less critical artifacts in regard to control complexity, a firm needs to determine the level of influence in the involved ecosystems. This factor is not explicitly covered by the CAP model and could be considered as an additional discussion point or as a separate decision factor in the contribution strategies which are elicited from the CAP model.

8.3 Direct and Indirect Use of OSS ecosystems

The second decision factor originating from the Kraljic's model [104] is the *profit impact*. Profit generally refers to the margin between what the customer is willing to pay for the final product and what the product costs to produce. For OSS ecosystems, this translates into how much *value* a firm can offer based on the OSS, e.g. services, and how much resources the firm needs to invest into integration and differentiation activities. I.e., much of the original definitions are preserved in the CAP model and the re-labeled decision factor business impact.

Artifacts with high profit, or high business impact are differential towards competitors and add significant value to the product and service offerings of the firm [120], e.g., the gaming services for Sony Mobile, see section 4.4. Analogous, artifacts with low profit are those related to commodity artifacts shared among the competitors, e.g., Linux commodity parts, see section 4.4. This reasoning works in cases where the OSS and its ecosystem is directly involved in the product or service which focal firm offers to its customers. The customers are those who decide which product to purchase, and therefore mainly contribute in the value creation process [9]. This requires good customer-understanding to judge which artifacts are the potential differentiators that will influence the purchase decision.

In cases where an OSS has an indirect relation to the product or service of the firm, the artifact's value becomes harder to judge. This is because the artifact may no longer have a clear connection to a requirement which has been elicited from a customer who is willing to pay for it. In these cases, firms need to decide themselves if a particular artifact gives them an advantage relative to its competitors.

OSS ecosystems often facilitates software engineering process innovations that later spark product innovations that increase the business impact of an artifact, e.g., if an artifact makes the development or delivery of the product to a higher quality or shorter time-to-market respectively [117]. These factors cannot be judged by marketing, but rather by the developers, architects and product managers who are involved on the technical aspects of software development and delivery. In regard to the CAP model, this indirect view of business impact may be managed by having a cross-functional mix of internal stakeholders and subject-matter experts that can help to give a complete picture of an artifact's business impact.

8.4 Comparing to Other Commoditization Models

Both commoditization models suggested by van der Linden et al. [120] and Bosch [22] consider how an artifact moves from a differential to a commoditized state. This is natural as technology and functionality matures and becomes standardized among actors on the same market or within the same OSS ecosystem. The impact of whether an artifact is to be considered differential or commodity is covered by the business impact factor of the CAP model. However, how quickly an artifact moves from one state to another is not explicitly captured by the CAP model. This

dimension requires firms to continuously use the CAP model and track the evolution of features and their business impact. We recommend that the evaluation is performed every time a new product is planned and use the reactive approach in combination with the proactive (see section 4.3 and 4.2 respectively).

Relative to the level of commoditization of an artifact, the two previous models consider how the artifact should be developed and shared. Van der Linden et al. [120] suggested to internally keep the differential artifacts and gradually share them as they become commoditized through intra-organizational collaborations and finally as OSS. In the CAP model, this aligns with the control complexity factor, i.e., how much control and influence is needed in regard to the artifact.

The main novelty of the CAP model in relation to the other commoditization models [22, 120] considers OSS ecosystem participation and enables improved synchronization towards firms' product strategy and product planning, via feature selection, prioritization and finally release planning [101]. The strategic aspect covered by the CAP model uses the commoditization principle together with business impact estimates and control complexity help may firms to better benefit from potential OI benefits. Assuming the commoditization is inevitable, the CAP model helps firms to fully benefit the business potential of differential features and timely share them with OSS ecosystems for achieving lower maintenance costs. Moreover, the CAP model helps to visualize the long term consequences of keeping or contributing an internally developed software artifact (more patches and longer time-to-market as consequence). Finally, the CAP model provides guidelines for how to position in an OSS ecosystem's governance structure [10] and how to influence it [41].

There may be various reasons why a firm would wish to contribute an artifact. Thus, the drivers used by Sony Mobile in the CAP model may not be the same for other firms wishing to adopt the model. The identified contribution drivers and cost structures should be aligned with the firm's understanding for how the value is drawn from the OSS ecosystems. This may help to improve the understanding of what should be contributed and how the resources should be planned in relation to these contributions. How the contribution objectives and drivers for contributions needs to be adapted is a topic for future research.

9 Conclusion

The recent changes in software business have forced software-intensive firms to rethink and re-plan the ways of creating and sustaining competitive advantage. The advent of OSS ecosystems has accelerated value creation, shortened time-to-market and reshaped commoditization processes. Harvesting these potential benefits requires improved support for strategic product planning in terms of clear guidelines of *what to develop internally* and *what to open up*. Currently available commoditization models [22,120] accurately capture the inevitability of com-

moditization in software business, but lack operational support that can be used to decide *what* and *when* to contribute to OSS ecosystems. Moreover, the existing software engineering literature lacks operational guidelines, for how software-intensive firms can formulate contribution strategies for improved *strategic product planning* at an artifact's level (e.g., features, requirements, test cases, frameworks or other enablers).

This paper introduces the Contribution Acceptance Process (CAP) which is developed to bridge product strategy with operational product planning and feature definition (**RQ1**). Moreover, the model is designed with commoditization in mind as it helps in setting contribution strategies in relation to the business value and control complexity aspects. Setting contribution strategies allow for *strategic product planning* that goes beyond feature definition, realization and release planning. The CAP model was developed in close collaboration with Sony Mobile that is actively involved in numerous OSS ecosystems. The model is an important step for firms that use these ecosystems in their product development and want to increase their OI benefits, such as increased innovation and shorter time-to-market. This paper also delivers an information meta-model that instantiates the CAP model and improves the communication and follow-up of current contribution strategies between the different parts of a firm, such as management, and development (**RQ2**).

There are several important avenues for future work around the CAP model. Firstly, we aim to validate the CAP model and related information meta-model in other firms, both statically and dynamically. We plan to focus on understanding the firm specific and independent parts of the CAP model. Secondly, we plan to continue to capture operational data from Sony Mobile and the three case firms related to the usage of the CAP model that will help in future improvements and adjustments. Thirdly, we plan to investigate how a contribution strategy can consider the influence a firm needs in an OSS ecosystems to be able to exercise control and introduce new features as needed. We believe that gaining and maintaining such influence in the right ecosystems is pivotal in order to execute successfully on contribution strategies. Fourthly, we want to investigate to what degree the CAP model supports innovation assessment for firms not working with OSS ecosystems. Our assumption is that these firms could use the CAP model to estimate the degree of innovativeness of the features (could be considered as an innovation benchmark) without setting contribution strategies. Lastly, we plan to explore which technical aspects should be considered and combined with the current strong business view of the CAP model (e.g. technical debt and architecture impact seems to be good candidates to be included).

CHAPTER V

OPEN TOOLS FOR SOFTWARE ENGINEERING USING THE THEORY OF OPENNESS: A VALIDATION STUDY IN THE AUTOMOTIVE INDUSTRY.

Abstract

Context: Open tools (e.g., Jenkins, Gerrit and Git) offer features or performance benefits that surpass their commercial counterparts. Many companies and developers from OSS communities create open tools in a collaborative effort in which software developers improve the code and share the changes within the community. We developed an empirically based theory for strategic choices on such tool development.
Aim: The aim of this study is to validate the theory of openness for tools in software engineering.
Method: We launched surveys in focus groups in two automotive industry companies and used the repertory grid technique to analyze the responses from participants in combination with qualitative data from discussions in the focus groups.
Results: We validated the theory in the *laggards* category (reactive, cost saving), as both companies belong to that category. Tools provided by suppliers are customized according to company needs to integrate into an already complex toolchain and thereby incurs expensive licenses. The lack of central tool coordination leads to multiple variants of the same tools, causing additional costs to glue tools together. Further, the lack of legal frameworks to work with OSS tools communi-

ties hampers companies to engage developers in OSS tools.
Conclusion: Both companies need a centralized, proactive strategy to help software developers use open standardized tools to reduce integration issues. It may require companies to foster an internal champion, which serves as an interface between the legal department, software developers and top management, to drive the open tools strategy.

1 Introduction

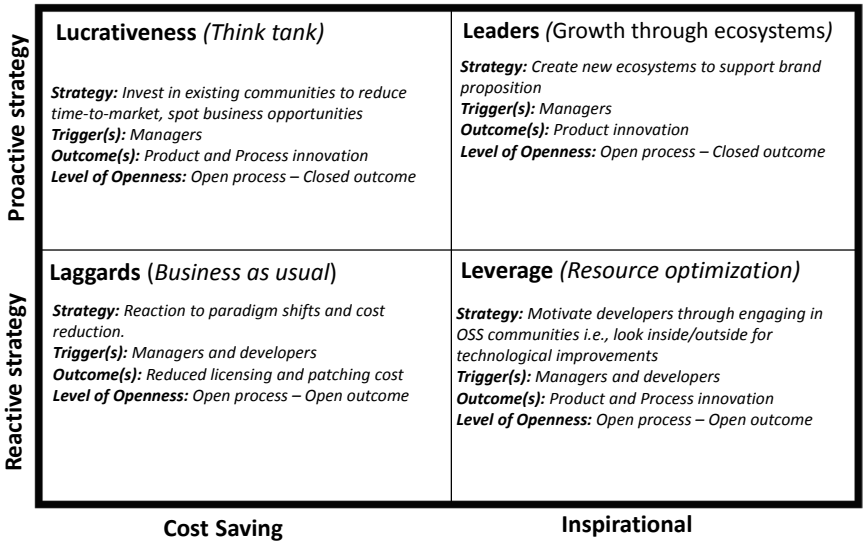
Using and co-developing open tools with other organizations for proprietary product development is becoming an increasingly open process, thanks to the ever-growing size of Open Source Software (OSS). Software-intensive companies create open tools communities and develop these tools with other companies to share the development and maintenance cost. For example, the Gerrit code review tool, built on top of the git version control system, was developed by Google and made open source in relation to Android development. Jenkins is another OSS tool co-developed by many companies (e.g., Sony Mobile, Ericsson, Intel, SAP etc) to make their continuous integration process faster and more flexible. Open tools may help companies to reduce the licensing costs for proprietary tools, and save development costs by offering flexibility in the development environment, increased turnaround speed, faster upgrades/releases and sharing the maintenance cost [137]. However, open tools require investment in the communities if companies would like to be leaders and gain influence on the development direction of these OSS tools [160].

This study continues our previous research efforts where we: 1) conducted a mapping study to identify the existing evidence on the use of open innovation in software engineering [141], 2) performed an exploratory case study at Sony about the use of open tools (e.g., Gerrit and Jenkins) in proprietary product development [137] and 3) conducted a survey in OSS tools communities (e.g., Gerrit, Jenkins, Git) and combined with the existing evidence to develop a theory of openness for tools in software engineering [140]. Figure 1 shows that the theory presents four categories of openness in companies with their respective focus:

1. Laggards – Routine business
2. Leverage – Resource optimization
3. Lucrativeness – Acting as a think-tank
4. Leaders – Growth through ecosystems

Each category has the different levels of openness, based on their strategies (proactive or reactive) in relation to goals (cost saving or inspirational). Typically, *laggards* respond to paradigm shifts and all strategies are reactive, aiming

Figure 1: Model of openness for tools



to reduce the development cost (i.e. integration). As for the *leverage* category, organizations use the external sources of innovation by inspiring their internal developers to participate in various OSS tools communities, prior to internal R&D work. It not only adds to product and process innovation but also inspires developers to exchange ideas on discussion forums to develop competence. *Lucrativeness* refers to investing in existing OSS communities to be able to influence and steer these communities in the same direction as the organizational interests. The purpose is to support internal innovation and reduce costs by investing in OSS tools communities. *Leaders* are organizations that focus on creating new communities and ecosystems to strengthen their business model. As a part of defining the theory of openness for tools, we formulated five proposition shown in table 1. The main goal of this study is to validate the theory of openness for tools in software engineering by conducting workshops in two automotive companies. Propositions derived from the theory of openness are validated based on the data collected from the workshop. The first two authors were directly involved in conducting the focus groups while the other two authors reviewed the collected data.

In this paper, we introduce related work on open source tools, as well as the analysis methods used here, in Section 2. Section 3 introduces the case companies, data collection procedures, validity threats, and analysis method to validate the propositions. Section 4 presents the analysis and discussion based on the repertory grid technique, using focus groups in the case companies. Finally, section 5 wraps up the study with the conclusion.

Table 1: Propositions from the theory [140]

ID	Proposition
P1	Openness of tools provides opportunities to reduce development costs.
P2	Openness of tools provides opportunities to shorten the development time.
P3	Openness of tools complements internal processes and product innovation.
P4	The degree of investment in OSS communities has an affect on the outcome.
P5	Introducing a proactive strategy, in relation to openness of tools, requires conscious management involvement.

2 Related work

Open Innovation (OI) builds on internal and external knowledge exchange that may or may not be associated with monetary transactions. OI can be realized as inside-out, outside- or coupled innovation processes [54]. OI has penetrated into several industries, as many companies discovered that their business may benefit by collaborating with OSS tools communities [30]. Companies apply OI in the area of OSS tools for company's internal product development. For example, our previous study [137] shows that Sony Mobile uses Jenkins and Gerrit ecosystems for knowledge flows between different companies' employees and Open Source Software (OSS) community developers, without monetary transactions. In contrast, companies like CloudBees monetize services related to the OSS tools. The stakeholders in the OSS tools communities not only share the innovation cost [27] and rewards, but also risks [137]. Companies may achieve that by revisiting their tool chains and look for open tools alternatives, as a natural and low-risk starting point in embracing openness.

The use of proprietary tools for software development leads to several challenges, e.g. delayed implementation of requirements, expensive licensing costs, inability of fixing things in-house, lack of customizability, and difficulty in finding solutions that meet current needs [137]. Companies use OSS tools communities to deal with several of these challenges [137]. However, in order to utilize open source tools communities for internal product development, companies need to invest their internal resources to use or contribute to OSS tools communites and have contribution strategies around it. The contribution strategies guides companies when to contribute and when to conceal in relation to their business model. The proposed theory of openness helps companies to choose the right level of openness while working with OSS tools communities.

Generating and acquiring data from software engineering activities is relatively

easy. The challenge is to use that data in a meaningful way to present something that is true rather than spurious. Theories offer common conceptual frameworks to summarize, condense and accumulate knowledge. SE research community have raised concerns on the lack of theories in SE [14, 20, 53, 80, 162, 169, 177] and pointed on the limited nature of the work about SE theories. We used Sjøberg et al's method to design the theory of openness [169] and here we use the repertory grid technique to validate the theory. Repertory grid is a technique for identifying the ways that a person interprets or gives meaning to his or her experience, being used in software engineering already. Moynihan's exploratory study used the repertory grid technique to identify the prevalent risks factors in software development projects. [134, 135]. Lee and Truex [111] used repertory grid technique to explore whether or not training in information system development methods improves students' cognitive structures (e.g., focused, tight thinking). Baddoo's exploratory study [12] collected data from 13 collaborating software companies to investigate how groups of staff in three designated roles (i.e. developers, project managers and senior managers) saw themselves and the other roles as being involved in software process improvement. Ghazi et al. [69] proposed a decision support method, using the repertory grid technique, which aids practitioners to choose the right level of exploratory testing (i.e. freestyle testing, a high degree of exploration, medium degree of exploration, a low degree of exploration and scripted) in their context.

3 Research methodology

Below, we describe the case companies, that data collection and analysis methods, and our discussion and actions taken to improve the validity of the study.

3.1 Case A

Company A is a major automotive industry manufacturer of heavy trucks and buses. It also manufactures diesel engines for heavy vehicles as well as marine and general industrial applications. The company employs approximately 42,100 people around the world. The company develops its software services for fleet management and is considering transport as a service perspective in their business model. Albeit their internally developed software is not monetized yet, the company has started seeing the software as an important strategic area for its core business. Therefore, the company has an abstract strategy of utilizing OSS tools to build competence in this area. The company uses Jenkins in the development of their fleet management system. The participants involved in the workshop were tools manager, product owners and technical teams leads and tools engineers.

3.2 Case B

Company B is one of the well-known global brands in automotive industry, heading towards an all-electric future. The company has 34,200 employees in five continents, and 2300 dealers globally. The increased use of software in the cars makes embedded software development a cutting-edge area for the company's business model. In order to facilitate the software development, the company buys tools to facilitate the software development from suppliers. However, it makes it difficult to buy a solution and fit it into the existing tool chain. Therefore, the company is shifting towards OSS tools to achieve standardization in all teams. Tools like Jenkins, Gerrit and Git are already utilized in the development process. The tool specially discussed in this study is an internally developed tool named *Awesome framework* by the company. It is an automated testing framework which enables test automation for hardware-in-the-loop, software-in-the-loop, model-in-the-loop test environments to be able to fit these into a continuous integration chain. It provides an integrated development environment for developing such tests at component and system level. The participants of the workshop in the case company include tools manager, product owners, business analysts, technical teams leads and tools engineers.

3.3 Data collection and analysis methods

Focus groups [159] were conducted at the two companies involved in this study. There were 12 and 10 participants in the focus groups conducted at the companies A and B, respectively. Furthermore, we used a repertory grid technique [95] to analyze and validate the theory of openness. Kelly proposed the personal construct theory (PCT) and the associated repertory grid technique in the 1950's to elicit and analyze personal constructs [95]. The idea behind the theory is that participants have their own view of the world based on their observation of surroundings. Therefore, each participant builds his own conceptual framework which leads to different opinions about the same problem. Participants constantly observe and react to their understanding of the surroundings. Consequently, participants reform their personal theories and assumptions [51].

Kelly's repertory grid technique is used to elicit, evaluate and analyze the constructs [95]. The grid is comprised of following three basic concepts: 1) Elements elicitation, 2) Constructs elicitation, 3) Ratings. *Elements* refer to individual aspects or objects of a topic, which participants try to understand. A *construct* is made of two contrasting concepts that are equally weighted on a bipolar scale. There are two essential ways to select grid elements: a) elicit elements from participants, b) provide participants with elements.

We used repertory grid to validate the theory of openness by providing the elements and constructs derived from the theory of openness [140]. Each elements was rated against each construct in the focus groups. To further support our choice, a number of studies have taken this approach [111, 143, 148, 199] by providing

elements and constructs to participants for the ratings. The description of elements, constructs and their ratings are as follow:

Elements derived from theory of openness: This study uses four elements from the theory of openness mentioned below:

1. Inspiration
2. Reactive
3. Cost saving
4. Proactive

The elements can be seen in figure 2 where we used the same elements for companies A and B.

Constructs derived from the theory of openness: We designed constructs from the theory of openness by creating the contrasting poles of each construct, see A1, A2 ... in Table 2. These constructs are derived from the existing literature in the theory of openness [140]. The constructs asses company's perspective on creating new communities to facilitate internal product development by reducing the development, development cost, new creation of roles, approval from top management for OSS tools adoption and access to skilled workforce etc.

Table 2: Constructs derived from theory of openness with their contrasting poles

ID	Similarity Pole (+)	Contrast Pole (-)
A1	Creating new open source communities to facilitate internal product development	Creating new open source communities not required to facilitate internal product development
A2	Creating new open source communities to explore emerging technologies	Creating new open source communities not required to explore emerging technologies
A3	Leveraging OSS to increase market share	Leveraging OSS to increase market share not required
A4	Using existing open source communities to identify the emerging technologies	Using existing open source communities to identify the emerging technologies not required
A5	Using existing open source communities for technological improvements	Using existing open source communities for technological improvements not required
A6	Investing in the open source communities to steer the community's development towards organizational benefits	Investing in the open source communities not required to steer the community's development towards organizational benefits

A7	Defining new work roles and teams to work with open source communities	Defining new work roles and teams not required to work with open source communities
A8	Adopting OSS tools need approval from top management	Adopting OSS tools does not need approval from top management
A9	Contributing to open source communities need approval from top management	Contributing to open source communities does not need approval from top management
A10	Educating the organization about OSS culture and ways of working	Educating the organization about OSS culture and ways of working not required
A11	Using OSS tools helps to reduce development time	Using OSS tools helps to reduce development time
A12	Working with open source communities gives access to free labor	Working with open source communities gives access to free labor
A13	Using OSS tools keeps developers updated with best tools practices	Using OSS tools keeps developers updated with best tools practices
A14	OSS tools reduces licensing cost in relation to proprietary tools	OSS tools reduces licensing cost in relation to proprietary tools
A15	The benefits of engaging in open source communities outweighs the risk of losing Intellectual property rights	The benefits of engaging in open source communities does not outweighs the risk of losing Intellectual property rights
A16	Engaging in open source communities provides fun ways of working for developers	Engaging in open source communities provides boring ways of working for developers
A17	Working with open source communities gives access to external knowledge from communities	Working with open source communities does not gives access to external knowledge from communities

Ratings: In this step, elements are then rated against each construct in the focus groups at companies A and B. We used a three-point Likert scale to rate each element against each construct, see Table 3.

First, the focus group participants in both companies were given an introduction of constructs and elements to develop a common understanding in the whole group. Second, participants from company A picked *Jenkins* and participants from company B selected an internal tool called *Awesome framework* for the focus

Table 3: Scale used for the ratings

	Scale		
Elements	**1**	**2**	**3**
Inspirational	Inspirational	Somewhat	No Inspirational
Reactive	Reactive	Somewhat	Not Reactive
Cost saving	Cost saving	Somewhat	Not Cost saving
Proactive	Proactive	Somewhat	Not Proactive

groups. Third, a survey link was distributed among all the participants to rate each element against the constructs, based on the selected tools from their internal development environment. Fourth, each element was rated against each construct using the scale in table 3.

For example, Figure 2 shows that adopting OSS tools need approval from top management **(A8)** is rated as not inspirational (3), reactive (1), not cost saving (3) and not proactive (3) by the participants in both companies. Similarly, the use of OSS tools helps to reduce the development time **(A11)** is rated as a cost saving (1) and proactive (1). Third, we held a discussion among participants based on the ratings to further explore their ratings. The discussion part was recorded and transcribed to further explore the rationales for the participant's ratings.

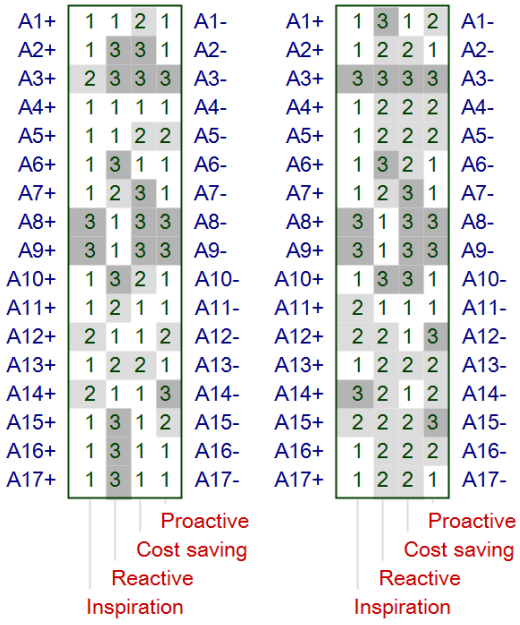
Outcome: The outcome of this step (ratings) is Figure 2, which shows ratings of focus groups from companies A and B.

3.4 Validity threats

We discuss validity threats with respect to internal validity, external validity, construct validity, and conclusion validity [159].

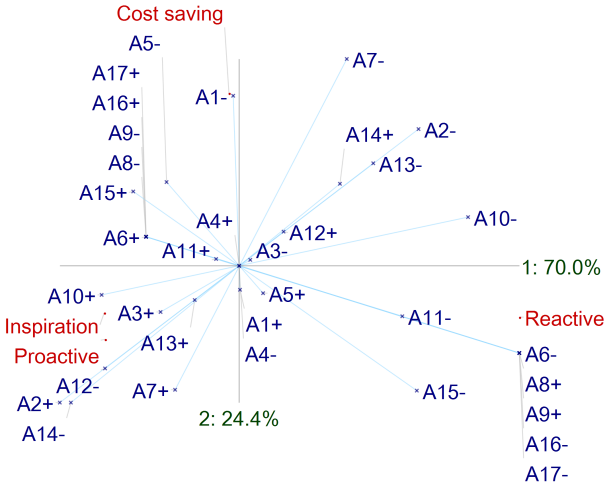
Internal validity: Internal validity threat may relate to confounding factors that may have influenced the outcome without researcher's knowledge. One possible threat is that the participants in companies A and B may have misunderstood the constructs and elements. Therefore, an introduction presentation was given to participants before rating each element against constructs. Constructs and elements were exemplified during the introduction presentation session. Furthermore, we also had a follow-up discussion based on the ratings from the participants to clarify misunderstandings. Another threat to internal validity was that the participants might had a difficulty in interpreting the scale for the ratings. To minimize the risk of scale's misunderstanding, the examples were given in the introduction prior to ratings of elements and constructs.

External validity: Both companies are experiencing a paradigm shift with ever increasing adoption of software in cars and trucks to stay on the competitive edge. Validating the theory in an automotive industry gives more weight to the external

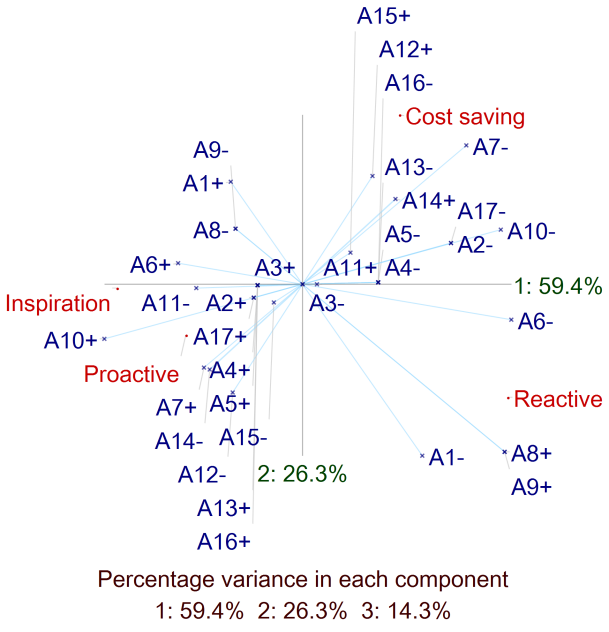


(a) Company A (b) Company B

Figure 2: Ratings of elements in relation to constructs from focus group participants, using the three point Likert scale presented in table 3.



(a) Company A



(b) Company B

Figure 3: Principal components analysis (PCA) of ratings from focus groups.

validity of the theory. However, automotive industry does not represent the whole software industry and therefore, the scope of the findings are limited. More studies are intended to perform in the software industry to improve the external validity of the theory.

Construct validity: Both constructs and elements in the theory are relevant to discuss in relation to construct validity. Neither of the case companies comes from an OSS development and community participation background and therefore, do not have established procedures to work with open tools communities. As a consequence, neither company has a well-defined procedure to map all the constructs of the theory. This threat was partially met by keeping the discussion on a higher level to the company's specific context.

Conclusion validity: First, participants in the focus groups may have tried guessing the propositions during the introduction presentation, which introduces a threat to the conclusion validity. This threat was mitigated by keeping elements and constructs as discrete as possible. Second, a researcher's bias while interpreting and recording focus groups mat threaten the conclusions. To reduce this threat, multiple researchers were involved in the focus groups (observer triangulation). Furthermore, focus groups were recorded, transcribed, and the results were validated by multiple researchers.

4 Results and discussion

The outcomes from the repertory grid technique are the grid and a PCA analysis. Figure 2 shows the formation of the repertory grid for companies A and B, based on the ratings from the participants in the focus group meetings. The vertical axis shows the contrasting poles of *constructs* derived from the theory of openness, mentioned in Table 2. The horizontal axis shows the *elements* derived from the theory of openness. Each element was rated by the participants in the workshop in relation to constructs using the scale presented in Table 3. Figure 3 shows the statistical repertory analysis based on the principal components analysis (PCA) [17]. PCA is a data reduction technique, used to find the dimensions of maximum variability in data. Figure 3 represents the spatial distances between and among the elements and constructs, and suggests how they might be related to each other.

4.1 Relation between strategies

The PCA shows that *proactive* and *inspiration* in companies A and B are spatially closer to each other. However, the *cost saving* and *reactive* strategies in company A seem to have a higher variance in contrast to company B. It can be explained by both companies' current strategy of buying OSS enterprise solutions instead of actively using or contributing to OSS tools communities, because they believe it gives them more control over the contract as the companies think in the old

way of subcontracting. The ratings in company A were not able to distinguish between OSS tools, and tools provided by suppliers based on OSS. The reason for this may be the lack of OSS understanding and competence at company A, since the company does not come from an OSS development background. One of the participants stated, *"one of the major reasons for not using OSS tools is the lack of competence and understanding. We do not know how to protect our intellectual property rights. We tend to go for buying solutions when there are intellectual property rights involved"*.

4.2 Validation of propositions

Further, we validate the five propositions (see table 1) from the theory of openness [140] in the light of the PCA and discussions in the focus groups.

For proposition **(P1)** about the reduction in development cost, the PCA shows that access to free labor **(A12+)** and reduced licensing cost **(A14+)** is closer to cost saving in case of company B as opposed to company A (see fig 3a and 3b). One possible explanation is that Company A does not see faster automated build servers as a competitive edge for its internal developed software. Furthermore, none of its software is provided as a service to the customers yet. However, they started seeing value in services and connections, where licensing costs might come into play. One of the participants stated that *"the percentage of money spent on licenses is actually not that high at the moment. However, if we start producing hundreds of micro-services with each of them need their own database, buying Oracle licenses might become really expensive"*. On the other hand, it is not uncommon for company B to buy tools and these tools can be expensive. One of the participants in company B stated that *"customized tools provided by big vendors like MathWorks for the company are really expensive. Therefore, we have a window of opportunity to update our environment with open tools."*. In conclusion, openness of tools has the potential to reduce the development cost, which validates our proposition **P1**.

Regarding whether openness shortens the development time **(P2)**, PCA for both companies in Figures 3a and 3b shows that the benefits of engaging in OSS tools communities outweigh the risk of losing IPRs **(A15+)**, and reduced development time **(A11+)** is spatially closer to cost saving. This indicates that participants think that engaging in open tool communities outweighs the risk of losing intellectual property rights, although the lack of maintenance support in the company is a point of concern for developers. Software developers in company B think that using open tools speed up their development in relation to seeking permission to buy a tool. One participant stated that *"regarding speed as a developer, it takes a few minutes to download an open framework rather than seeking an approval from the manager to buy a tool, which takes forever"*. However, both companies are careful in selecting and integrating these open tools to make sure that these tools won't die over time. There is a risk that software developers might not get the required funding and enough management backing to maintain these open tools internally

if the community dies. A participant stated that *"we are dependent on some really large python projects that we know won't die anytime soon."*. Therefore, the proposition **P2** about open tools may reduce the development is confirmed by the data.

In relation to process and product innovation **(P3)**, it should be mentioned we have earlier observed that process innovation leads to product innovation [117]. However, neither case company considers tools front leading innovation in the development of their core products. In particular, the tools development team of company B does not get enough funding and management backing, although other departments in the company need tool support to facilitate product innovation in the core products. This is a general problem with the project focused companies, where the budget is allocated for the given project only, with limited consideration for long-term investment. The tools department does not have an allocated budget for the development of tools and the maintenance of these to achieve standardization. One of the participants in company B stated *"when we buy tools, it is part of the R&D budget and there is no designated budget for maintenance. For example, if we get the funding for developing an internal tool like Awesome framework, there is no allocation of budget for the maintenance of that tool"*. On the other hand, both companies identify the need for process innovation by creating new roles [129] to deal with the challenges of open tools platform. Consequently, there have been initiatives in both companies to create new open innovation roles. For example, a participant of company A mentioned that *"the company has actually employed a new role of open source cloud manager to create procedures and guidelines regarding how to be active in an open space"*. Furthermore, a participant in company B stated that *"there is a new initiative called Nordic foundation partly driven by Ericsson to spread open source usage and with the already formed legal framework among Nordic companies. It could be a fast track for us but it needs to be championed by R&D in order to reap benefits"*. Therefore, the proposition **P3** related to the openness of tools complements internal process and product innovation is confirmed by the data.

In both companies, PCA shows that adopting **(A8+)** and contributing to OSS tools **(A9+)** are spatially close to the *reactive* strategy, meaning that participants consider taking approval from top management in either adopting or contributing to open tools communities as a reactive strategy. One possible explanation for seeking approval is that neither company has a legal framework to work with open tools communities. One of the participants in company A stated that *"we have a top-level statement saying* Open Source First *but what does that exactly mean? It does not state exactly what, why and how should we go open and how these decisions are supported"*. Similarly, a participant in company B also stated that *"we do not have any internal procedure or legal framework to create new or work with the existing open tools communities"*. Therefore, **P4** about the relation between investment in OSS tools communities and its impact, is neither confirmed nor rejected by the data, since both case companies lack internal procedures to invest

their internal resources in open tools communities.

Finally, proposition **P5**, about the role of management in Table 1, needs to be extended since not only the proactive strategy but also the reactive strategy, requires management involvement, as software developers need a legal framework to work with open tools.

4.3 Cross analysis with Sony Mobile

After the focus group meeting, we conducted a cross case analysis with our previous Sony Mobile case [137]. That study explored Sony Mobile's transition from closed tools to open tools platform [137], which constitutes one of the cases for an empirical grounding of the theory of openness for tools. Therefore, in this cross case analysis, we have found three similar patterns in companies A and B in relation to Sony Mobile.

First, there was a paradigm shift when Sony Mobile moved from Windows to Linux, which in turn required Sony to move from proprietary tools (e.g, ElectricCommander) to open tools platforms (e.g., Gerrit, Jenkins, Git). The complex integration tool chain triggered the move towards open tools platforms and it resulted in increased development speed and reduced time to market. Both companies A and B are experiencing a paradigm shift in the automotive industry, where software is becoming the key area for the development of vehicles such an autonomous driving features and providing transportation as a service. Both companies are very reactive in their approach when opting for open tools. A participant in the company said, *"the change generally depends on how painful the situation is. A few years ago, we had 10 different teams working with an internal platform using different tool chains and had a great difficulty in glowing the tool chain together. We hired an external company that glued our tools chain together with massive scripts using an open tool solution"*. Therefore, the open tools chosen in a proactive way of standardizing the tool chain in all teams, may play an important role in the development of software-intensive vehicles.

Second, when Sony Mobile realized that they lacked legal procedures to work with open tools communities, they created an internal open tools department consists of legal experts, developers and managers to create a well-established procedure for using or contributing to open tools platforms. Companies A and B may follow the same steps by creating an internal process to educate the developers and develop competence to work with open tools communities. As one of the participants in company A said, *"we lack competence and streamlined decisions on how to work with open tools communities"*. Another participant in company B said that *"we need improvements in the areas of governance, training and policies."*

Third, Sony Mobile had an internal champion to derive the whole open toolchain movement with the support of software developers and legal team. This played an important role in convincing the top management to support and fund open tools platforms, however, it took three years for Sony Mobile to make it. The

current case companies may need a similar, internal champion to act as an interface between the legal team, software developers, and the top management.

Key takeaway: Companies A and B may choose a centralized proactive strategy similar to Sony Mobile (reactive in that case) to make internal legal policies and education for employees to meet the ever-growing size of software in automobiles.

4.4 Business implications

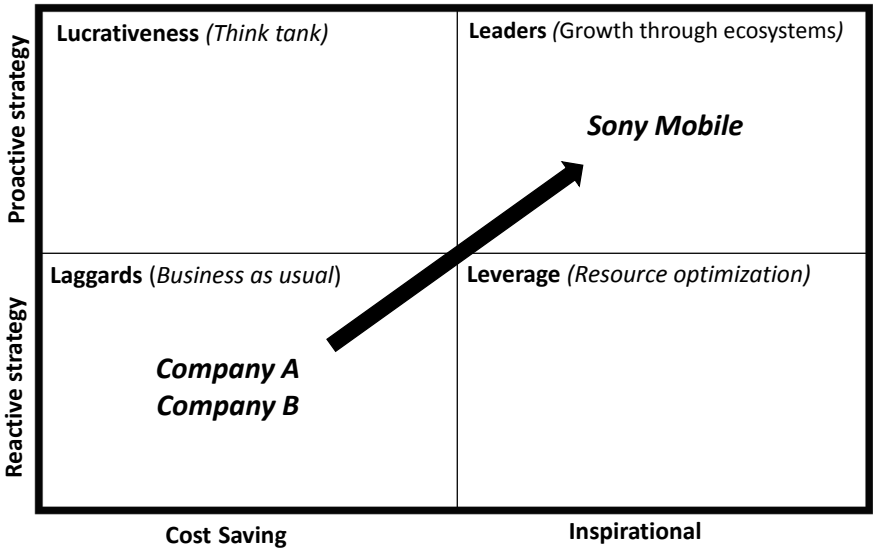
Automotive industry sees software services as a new emerging area in terms of transportation as a service that will be software-enabled. This pattern can be related to Sony and Android case in which natural drive for saving money and reducing ownership costs was to create industry wide standard platforms and Google capitalized on the first mover advantage by offering Android for free and it got widely adopted by vendors. Initially, Sony underestimated the value in services and apps. So everyone let Google develop Android and occupy services such as Gmail, drive, gaming, camera etc. Now Android is provided as an empty shell and vendors just provide devices to use services for money. Similar risk may apply to automotive industry, if the companies do not own their code base by establishing OSS communities and ecosystems. Companies may provide vehicles to run the services and make money on the services instead of selling the vehicles.

4.5 Implications for theory of openness

Figure 4 shows the mapping of companies A and B on the model of openness for tools [140]. Both companies qualify as **laggards** since they have no internal procedures to facilitate developers to contribute to OSS tools communities. Both companies react when the integration of tools becomes painful for developers, rather than standardize the ways of working in the companies. Also for laggards, we noticed that it is important to have the management support and approval. In case company B, the internally developed tool named *Awesome framework* is meant to be shared among different partners of the company and it is used in core product development teams. However, the development does not have enough resources to support core product development teams who have requirements and need support. This suggests the importance of the development of the tools but lack of funding and resources hampers the work of core product development teams who are dependent on the tool.

The arrow in figure 4 shows that Sony Mobile went through the same transition as companies A and B are going through. Sony Mobile started as a **laggard** with its complex continuous integration tools chain, before actively using and contributing to OSS tools communities like Jenkins and Gerrit. However, this transition required Sony Mobile to create a tools department with the legal framework to utilize the OSS tools communities.

Figure 4: Mapping of companies on the model of openness for tools.



In the case of two studied companies, company B is found to be more in need of this transition due to their complex integration tool chain compared to company A. Therefore, both companies may have processes and legal frameworks for developers, to encourage them to take more initiative towards open tools in order to standardize the ways of working. Currently, in the company B, all teams have their internally developed tools with all possible technologies (e.g., C#, Python, Java etc.) which leads to loss of resources on writing more glue code to integrate the tool chain. A participant from company B stated, *"I think we have to change our way of thinking and we never share anything because there is always some kind of minor internal change that drives our testing and we do not want to share that. However, 80% of the rest is the same that can be shared. At the moment, it is all or nothing and we need to calm down a little bit and a share the common parts atleast"*.

Once the processes for contribution and using OSS tools communities are in place, both companies may consider becoming new open tools ecosystems **(leaders)** by making the code open, to attract other manufactures in the industry with the same needs. One of the participants stated, *"I think it is a question of survival in the near future and we do not have time to deal with the increase in the number of variants. It will be nice to replace the closed source tools with the open ones so that we can actually fix tools without glue scripts"*. It is to be noted that the data collected from the case companies does not address or validate the lever-

age and lucrativeness category in Fig 4. The studied companies only validates the laggards category, thus the theory required more validation studies to validate the lucrativeness and leverage category as well.

Key takeaway: Both companies may learn from Sony Mobile's transition from **laggards** to **leaders** by innovating their process in terms of creating a legal framework. The framework will help companies to engage their developers in OSS tools communities together with the legal team to eventually be able to build new communities (**leaders**) to facilitate their core product development. However, **laggards** also require management to look at tools as a long term standardization of processes perspective rather than a project based approach.

5 Conclusions

This study aims to validate our theory of openness for software engineering tools. We use the repertory grid technique to discuss five propositions derived from the theory of openness and presents a cross analysis with Sony Mobile. We were able to validate the theory for the laggards category, with two companies from the automotive domain.

The findings suggest that both case companies lack internal procedures to work with the open tools communities. This leads to frustration among employees for not knowing why and how to work with open tools communities. It might be because both companies come from a closed background of manufacturing cars and trucks. However, both companies are currently experiencing a paradigm shift in the automotive industry with more software introduced in the vehicles. This may require the management to rethink and revise their strategy to extract the external knowledge by using open tools communities.

The strategy should address creating a new role which works as an interface between legal department, software developers and management like Sony Mobile. Furthermore, it may entail creating processes regarding how and when to use, contribute or create new open tools platforms for company's benefits. As for future work, more validation studies need to be conducted to validate the propositions derived from the theory of openness.

BIBLIOGRAPHY

[1] CVSAnalY by MetricsGrimoire. Accessed: 2014-07-17.

[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] Pär J Ågerfalk and Brian Fitzgerald. Outsourcing to an unknown workforce: Exploring opensurcing as a global sourcing strategy. *MIS quarterly*, pages 385–409, 2008.

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

[7] 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.

[8] Marnix Assink. Inhibitors of disruptive innovation capability: a conceptual model. *European Journal of Innovation Management*, 9(2):215–233, 2006.

[9] Aybüke Aurum and Claes Wohlin. A value-based approach in requirements engineering: Explaining some of the fundamental concepts. pages 109–115, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[10] Alfred Baars and Slinger Jansen. A framework for software ecosystem governance. In *International Conference of Software Business*, pages 168–180. Springer, 2012.

[11] Deepika Badampudi, Claes Wohlin, and Kai Petersen. Software component decision-making: In-house, oss, cots or outsourcing - a systematic literature review. *Journal of Systems and Software*, 121:105 – 124, 2016.

[12] Nathan Baddoo and Tracy Hall. Practitioner roles in software process improvement: an analysis using grid technique. *Software Process: Improvement and Practice*, 7(1):17–31, 2002.

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

[14] Victor R Basili, Forrest Shull, and Filippo Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473, 1999.

[15] 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.

[16] Willem Bekkers, Inge van de Weerd, Marco Spruit, and Sjaak Brinkkemper. A framework for process improvement in software product management. In *European Conference on Software Process Improvement*, pages 1–12. Springer, 2010.

[17] Richard C Bell. Analytic issues in the use of repertory grid technique. *Advances in personal construct psychology*, 1:25–48, 1990.

[18] Fiona Beyer and Kath Wright. Comprehensive searching for systematic reviews: a comparison of database performance. *Centre for Reviews and Dissemination, University of York*, 2011.

[19] Christian Bird and Nachiappan Nagappan. Who? where? what?: Examining distributed development in two large open source projects. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, MSR '12, pages 237–246, USA, 2012. IEEE Press.

[20] Elizabeth Bjarnason, Kari Smolander, Emelie Engström, and Per Runeson. A theory of distances in software engineering. *Information and Software Technology*, 70:204–219, 2016.

[21] Elizabeth Bjarnason, Michael Unterkalmsteiner, Emelie Engström, and Markus Borg. An industrial case study on test cases as requirements. *International Conference on Agile Software Development*, pages 27–39, 2015.

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

[23] Jan Bosch. Speed, data, and ecosystems: The future of software engineering. *IEEE Software*, 33(1):82–88, 2016.

[24] Stelian Brad, Mircea Fulea, Bogdan Mocan, Adina Duca, and Emilia Brad. Software platform for supporting open innovation. In *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, volume 3, pages 224–229. IEEE, 2008.

[25] Marjolein Caniëls and Cees Gelderman. Purchasing strategies in the kraljic matrix a power and dependence perspective. *Journal of Purchasing and Supply Management*, 11(2 - 3):141 – 155, 2005.

[26] Jenny M Carroll and Paul A Swatman. Structured-case: a methodological framework for building theory in information systems research. *European Journal of Information Systems*, 9(4):235–242, 2000.

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

[28] Henry Chesbrough and Sabine Brunswicker. A fad or a phenomenon?: The adoption of open innovation practices in large firms. *Research-Technology Management*, 57(2):16–25, 2014.

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

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

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

[32] Henry William Chesbrough and Melissa M. Appleyard. Open innovation and strategy. *California Management Review*, 50(1):57–76, 2007.

[33] 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.

[34] 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.

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

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

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

[38] 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.

[39] Linus Dahlander and David M. Gann. How open is innovation? *Research Policy*, 39(6):699 – 709, 2010.

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

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

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

[43] Daniela Damian. Stakeholders in global requirements engineering: Lessons learned from practice. *Software, IEEE*, 24(2):21–27, 2007.

[44] Sherae Daniel, Likoebe Maruping, Marcelo Cataldo, and James Herbsleb. When cultures clash: Participation in open source communities and its implications for organizational commitment. *Proceedings of the International Conference on Information Systems, Shanghai, China*, 2011.

[45] 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.

[46] 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.

[47] 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.

[48] Jingshu Du, Bart Leten, Wim Vanhaverbeke, and Henry Lopez-Vega. When research meets development: antecedents and implications of transfer speed. *Journal of Product Innovation Management*, 31(6):1181–1198, 2014.

[49] Winfried Ebner, Jan Marco Leimeister, and Helmut Krcmar. Community engineering for innovations: the ideas competition as a method to nurture a virtual community for innovations. *R&D Management*, 39(4):342–356, 2009.

[50] 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.

[51] Helen M Edwards, Sharon McDonald, and S Michelle Young. The repertory grid technique: Its place in empirical software engineering research. *Information and Software Technology*, 51(4):785–798, 2009.

[52] K El-Emam, DC Hoaglin, PW Jones, BA Kitchenham, SL Pfleeger, LM Pickard, and J Rosenberg. Preliminary guidelines for emprical research in software engineering. *National Research Council of Canada*, 2001.

[53] Albert Endres and H Dieter Rombach. *A handbook of software and systems engineering: Empirical observations, laws, and theories*. Pearson Education, 2003.

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

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

[56] Frederick D. Erickson. Qualitative methods in research on teaching. In Merlin C. Wittrock, editor, *Handbook of research on teaching*, pages 119–161. MacMillan, New York, NY, 3rd edition, 1986.

[57] 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.

[58] Daniel Méndez Fernández, Birgit Penzenstadler, Marco Kuhrmann, and Manfred Broy. A meta model for artefact-orientation: Fundamentals and lessons learned in requirements engineering. *Lecture Notes in Computer Science*, pages 183–197, 2010.

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

[60] Bent Flyvbjerg. *Five Misunderstandings about Case-Study Research*. SAGE, concise paperback edition, 2007.

[61] 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.

[62] Samuel Fricker. Software product management. In *Software for People*, pages 53–81. Springer, 2012.

[63] G.R. Gangadharan, Lorna Uden, and Paul Luttighuis. Sourcing Requirements and Designs for Software as a Service. *International Journal of Systems and Service-Oriented Engineering*, 6(1):1–16, jan 2016.

[64] Francisco José García-Peñalvo and Alicia García-Holgado. *Open Source Solutions for Knowledge Management and Technological Ecosystems*. IGI Global, 2017.

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

[66] Oliver Gassmann and Ellen Enkel. Towards a theory of open innovation: three core process archetypes. In *Proceedings of the R&D Management Conference*, pages 1–18, Portugal, 2004.

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

[68] Cees Gelderman and Arjan Weele. Handling measurement issues and strategic directions in kraljic's purchasing portfolio model. *Journal of Purchasing and Supply Management*, 9(5 - 6):207 – 216, 2003.

[69] Ahmad Nauman Ghazi, Kai Petersen, Claes Wohlin, and Elizabeth Bjarnason. A decision support method for recommending degrees of exploration in exploratory testing. *arXiv preprint arXiv:1704.00994*, 2017.

[70] Jesus M Gonzalez-Barahona, Daniel Izquierdo-Cortazar, Stefano Maffulli, and Gregorio Robles. Understanding how companies interact with free software communities. *IEEE software*, 30(5):38–45, 2013.

[71] Tony Gorschek, Per Garre, Stig Larsson, and Claes Wohlin. A model for technology transfer in practice. *IEEE software*, 23(6):88–95, 2006.

[72] Trisha Greenhalgh and Richard Peacock. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *Bmj*, 331(7524):1064–1065, 2005.

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

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

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

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

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

[78] 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.

[79] Joachim Henkel, Simone Schöberl, and Oliver Alexy. The emergence of openness: How and why firms adopt selective revealing in open innovation. *Research Policy*, 43(5):879–890, 2014.

[80] James D Herbsleb and Audris Mockus. Formulation and preliminary test of an empirical theory of coordination in software engineering. In *ACM SIGSOFT Software Engineering Notes*, volume 28, pages 138–137. ACM, 2003.

[81] Alan Hevner, Salvatore March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, March 2004.

[82] Martin Höst, Klaas-Jan Stol, and Alma Oručević-Alagić. *Inner source project management*. Springer, 2014.

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

[84] Stefan Hüsig and Stefan 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.

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

[86] 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.

[87] Slinger Jansen, Sjaak Brinkkemper, and Anthony Finkelstein. Business network management as a survival strategy: A tale of two software ecosystems. *Proccedings of the 1st International Workshop on Software Ecosystems*, pages 34–48, 2009.

[88] 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.

[89] 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, July 2012.

[90] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. A sense of community: A research agenda for software ecosystems. In *31st International Conference on Software Engineering*, pages 187–190. IEEE, 2009.

[91] Chris Jensen and Walt Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *29th International Conference on Software Engineering (ICSE'07)*, pages 364–374, May 2007.

[92] Chris Jensen and Walt Scacchi. Governance in open source software development projects: A comparative multi-level analysis. In *IFIP International Conference on Open Source Systems*, pages 130–142. Springer, 2010.

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

[94] Lena Karlsson, 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.

[95] George A Kelly. *The psychology of personal constructs. I. A theory of personality. II. Clinical diagnosis and psychotherapy.* Norton & Co., 1955.

[96] Richard Kemp. Open source software (oss) governance in the organisation. *Computer Law & Security Review*, 26(3):309–316, 2010.

[97] 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.

[98] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15, 2009.

[99] 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.

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

[101] lccn=2008939365 year=2008 publisher=Springer Berlin Heidelberg Kittlaus, Hans-Bernd and Clough, Peter N, isbn=9783540769873. *Software Product Management and Pricing: Key Success Factors for Software Organizations*.

[102] 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.

[103] Marko Komssi, Marjo Kauppinen, Harri Töhönen, Laura Lehtola, and Alan Davis. Roadmapping problems in practice: value creation from the perspective of the customers. *Requirements Engineering*, 20(1):45–69, 2015.

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

[105] Harold L Kundel and Marcia Polansky. Measurement of observer agreement. *Radiology*, 228(2):303, 2003.

[106] 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.

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

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

[109] Keld Laursen and Ammon Salter. Open for innovation: the role of openness in explaining innovation performance among uk manufacturing firms. *Strategic management journal*, 27(2):131–150, 2006.

[110] 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.

[111] Jungwoo Lee and Duane P Truex. Exploring the impact of formal training in isd methods on the cognitive structure of novice information systems developers. *Information Systems Journal*, 10(4):347–367, 2000.

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

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

[114] Ulrich Lichtenthaler and Holger Ernst. Opening up the innovation process: the role of technology aggressiveness. *R&d Management*, 39(1):38–54, 2009.

[115] 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.

[116] Johan Linåker, Maria Krantz, and Martin Höst. On infrastructure for facilitation of inner source in small development teams. In *Product-Focused Software Process Improvement*, pages 149–163. Springer, 2014.

[117] 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.

[118] Johan Linåker, Hussan Munir, Krzysztof Wnuk, and Carl-Eric Mols. Motivating the contributions: An open innovation perspective on what to share as open source software. *Journal of Systems and Software*, 135(Supplement C):17 – 36, 2018.

[119] Johan Linåker, Patrick Rempel, Björn Regnell, and Patrick Mäder. How firms adapt and interact in open source ecosystems: Analyzing stakeholder influence and collaboration patterns. In *Requirements Engineering: Foundation for Software Quality*, pages 63–81. Springer, 2016.

[120] 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.

[121] 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.

[122] Andrey Maglyas and Samuel Fricker. The preliminary results from the software product management state-of-practice survey. In Casper Lassenius and Kari Smolander, editors, *International Conference on Software Business*, pages 295–300, Paphos, Cyprus, 2014.

[123] Konstantinos Manikas and Klaus Hansen. Software ecosystems–a systematic literature review. *Journal of Systems and Software*, 86(5):1294–1306, 2013.

[124] Daniel Méndez Fernández, Stefan Wagner, Klaus Lochmann, Andrea Baumann, and Holger de Carne. Field study on requirements engineering: Investigation of artifacts, project parameters, and execution strategies. *Information and Software Technology*, 54(2):162–178, 2012.

[125] Sharan Merriam. What can you tell from an N of 1?: Issues of validity and reliability in qualitative research. *PAACE Journal of Lifelong Learning*, 4:50–60, 1995.

[126] Robert King Merton. *Social theory and social structure*. Simon and Schuster, 1968.

[127] Audris Mockus and James D. Herbsleb. Why not improve coordination in distributed software development by stealing good ideas from open source. In *Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering*, pages 19–25, 2002.

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

[129] Carl-Eric Mols, Krzysztof Wnuk, and Johan Linåker. The open source officer role – experiences. In Federico Balaguer, Roberto Di Cosmo, Alejandra Garrido, Fabio Kon, Gregorio Robles, and Stefano Zacchiroli, editors, *Open Source Systems: Towards Robust Practices*, pages 55–59, Cham, 2017. Springer International Publishing.

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

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

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

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

[134] Tony Moynihan. An inventory of personal constructs for information systems project risk researchers. *Journal of information technology*, 11(4):359–371, 1996.

[135] Tony Moynihan. How experienced project managers assess risk. *IEEE software*, 14(3):35–41, 1997.

[136] Neeshal Munga, Thomas Fogwill, and Quentin Williams. The adoption of open source software in business models: A Red Hat and IBM case study. *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 112–121, 2009.

[137] Hussan Munir, Johan Linåker, Krzysztof Wnuk, Per Runeson, and Björn Regnell. Open innovation using open source tools: A case study at Sony Mobile. *Empirical Software Engineering*, pages 1–38, 2017.

[138] Hussan Munir, Misagh Moayyed, and Kai Petersen. Considering rigor and relevance when evaluating test driven development: A systematic review. *Information and Software Technology*, 56(4):375 – 394, 2014.

[139] Hussan Munir and Per Runeson. Software testing in open innovation: An exploratory case study of the acceptance test harness for Jenkins. *International Conference on Software and Systems Process 2015*, pages 187–191, 2015.

[140] Hussan Munir, Per Runeson, and Krzysztof Wnuk. A theory of openness for software engineering tools in software organizations. *Information and Software Technology*, 97:26 – 45, 2018.

[141] Hussan Munir, Krzysztof Wnuk, and Per Runeson. Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering*, 21(2):684–723, Apr 2016.

[142] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. *Evolution patterns of open-source software systems and communities*. ACM, 2002.

[143] Nan Niu and Steve Easterbrook. Discovering aspects in requirements with repertory grid. In *Proceedings of the 2006 international workshop on Early aspects at ICSE*, pages 35–42. ACM, 2006.

[144] 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.

[145] Vinit Parida, Mats Westerberg, and Johan Frishammar. Effect of open innovation practices on SMEs innovative performance: An empirical study. *ICSB World Conference Proceedings*, page 1, 2011.

[146] 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*, pages 351–354, 2011.

[147] 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.

[148] Gary John Phythian and Malcolm King. Developing an expert support system for tender enquiry evaluation: A case study. *European Journal of Operational Research*, 56(1):15–29, 1992.

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

[150] 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.

[151] Miikka Poikselkä, Harri Holma, Jukka Hongisto, Juha Kallio, and Antti Toskala. *Voice over LTE (VoLTE)*. John Wiley & Sons, 2012.

[152] Jennie Popay, Helen Roberts, Amanda Sowden, Mark Petticrew, Lisa Arai, Mark Rodgers, Nicky Britten, Katrina Roen, and Steven Duffy. Guidance on the conduct of narrative synthesis in systematic reviews. *A product from the ESRC methods programme Version*, 1:b92, 2006.

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

[154] Björn Regnell and Sjaak Brinkkemper. Market-driven requirements engineering for software products. In *Engineering and managing software requirements*, pages 287–308. Springer, 2005.

[155] Colin Robson and Kieran McCartan. *Real World Research*. John Wiley & Sons, January 2016.

[156] 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.

[157] 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.

[158] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.

[159] Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, March 2012.

[160] Per Runeson, Hussan Munir, and Krzysztof Wnuk. It is more blessed to give than to receive–open software tools enable open innovation. *Tiny Transactions on Computer Science*, 4, 2015.

[161] 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.

[162] Chris Sauer, D Ross Jeffery, Lesley Land, and Philip Yetton. The effectiveness of software development technical reviews: A behaviorally motivated program of research. *IEEE Transactions on Software Engineering*, 26(1):1–14, 2000.

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

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

[165] William R.. Shadish, Thomas D Cook, and Donald Thomas Campbell. *Experimental and quasi-experimental designs for generalized causal inference*. Wadsworth Cengage learning, 2002.

[166] Bashar Shaya. Process handling: A study for optimizing the processes for sourcing it and managing software licenses. *Master Thesis Industrial Economics and Management (Dept.), KTH, Sweden*, 2012.

[167] Janice Singer and Norman G. Vinson. Ethical issues in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 28(12):1171–1180, 2002.

[168] 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 - Proceedings of the 4th International Workshop on Social Software Engineering, pages 1–5, 2011.

[169] Dag Sjøberg, Tore Dybå, Bente CD Anda, and Jo E Hannay. Building theories in software engineering. In *Guide to advanced empirical software engineering*, pages 312–336. Springer, 2008.

[170] Dag 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.

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

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

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

[174] Klaas-Jan Stol, Paris Avgeriou, Muhammad Ali Babar, Yan Lucas, and Brian Fitzgerald. Key factors for adopting inner source. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 23(2):18, 2014.

[175] Klaas-Jan Stol, Michael Goedicke, and Ivar Jacobson. Introduction to the special section - general theories of software engineering: New advances and implications for research. *Information and Software Technology*, 70:176 – 180, 2016.

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

[177] Walter F Tichy. Should computer scientists experiment more? *Computer*, 31(5):32–40, 1998.

[178] Pauliina Ulkuniemi, Luis Araujo, and Jaana Tähtinen. Purchasing as market-shaping: The case of component-based software engineering. *Industrial Marketing Management*, 44:54 – 62, 2015.

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

[180] 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.

[181] Georg Von Krogh and Sebastian Spaeth. The open source software phenomenon: characteristics that promote research. *Journal of Strategic Information Systems*, 16(3):236 – 53, 2007.

[182] 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.

[183] 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.

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

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

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

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

[188] 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.

[189] Joel West and Scott Gallagher. Patterns of open innovation in open source software. *Open Innovation: researching a new paradigm*, 235(11), 2006.

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

[191] 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.

[192] Jim Whitehurst. *The Open Organization*. Harvard Business Review Press, 2015.

[193] 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.

[194] Krzysztof Wnuk, Dietmar Pfahl, David Callele, and Even Karlsson. How can open source software development help requirements management gain the potential of open innovation: an exploratory study. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 271–280. ACM, 2012.

[195] Krzysztof Wnuk, Björn Regnell, and Brian Berenbach. Scaling up requirements engineering - exploring the challenges of increasing size and complexity in market-driven software development. *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 54–59, 2011.

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

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

[198] Robert K Yin. Case study research: design and methods, applied social research methods series. *Thousand Oaks, CA: Sage Publications, Inc. Afacan, Y., & Erbug, C.(2009). An interdisciplinary heuristic evaluation method for universal building design. Journal of Applied Ergonomics*, 40:731–744, 2003.

[199] S Michelle Young, Helen M Edwards, Sharon McDonald, and J Barrie Thompson. Personality characteristics in an xp team: a repertory grid study. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 1–7. ACM, 2005.

[200] 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.