Towards Strategic Support for Requirements Engineering in Open Source Software Ecosystems

- What to reveal, when and to whom?

Johan Linåker

Licentiate Thesis, 2016
Department of Computer Science
Lund University

LU-CS-DISS: 2016-2
Licentiate Thesis 2, 2016
ISSN: **1652-4691**

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

Email: johan.linaker@cs.lth.se
WWW: `http://cs.lth.se/johan_linaker`

Printed in Sweden by Tryckeriet i E-huset, Lund, 2016

ABSTRACT

Background: Open Source Software (OSS) today makes up a central part and source of innovation for a diversity of firms and their business models. This is done either directly as a part of, or indirectly as an enabler for their product and service offerings. The openness implies a membership for the focal firm to a dynamic ecosystem with both known and unknown stakeholders. This needs consideration in regards to how the firm should act and bridge their internal Requirements Engineering (RE) process with the external one, and in the end what to share openly with ecosystem, when and to whom.
Aim: The aim is to create a foundation for future strategic support that can help firms involved in OSS ecosystems to make decisions on what to share and what to conceal, in order to influence the external RE process to align with internal strategies and business incentives.
Research Methodology: To investigate the problem from a real-world perspective, empirical software engineering research methods with a focus on case studies are used. Studies were performed both from a firm as well as an ecosystem perspective to understand the interaction between them. Open Innovation theory is used to define the problem from a firm's perspective.
Results: The studies suggest that firms adopting OSS internally can use the external workforce of the OSS ecosystem as a source of innovation in regards to both their internal processes and products. RE processes towards the ecosystem are informal and social, matching the general culture of OSS RE. Software considered as non-competitive or commodity are shared openly while stricter guidelines apply for that which has a higher business criticality or need for control. The OSS ecosystems in which the firms operate have evolving stakeholder populations where firms' influence and collaboration fluctuates with time. Influence is a pertinent attribute in stakeholder identification and analysis of OSS ecosystems. It can help understand stakeholders' agendas and provide input to contribution decisions.
Conclusion: By creating guidelines for what to share, when and to whom in OSS ecosystems, firms can align and bridge internal strategies and RE process with the ecosystems'. Future strategic support should combine such guidelines with the input of a systematic and continuous stakeholder analysis process of the ecosystems in terms of the stakeholders' influence and interactions in the ecosystems.

ACKNOWLEDGEMENTS

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

I would like to express my deepest gratitude to everyone, without whose help this licentiate thesis would not be possible. Special thanks goes out to Prof. Dr. Björn Regnell and Prof. Dr. Per Runeson for providing excellent and comforting guidance throughout this process. Thanks to Dr. Krzysztof Wnuk for rewarding and entertaining collaborations. Thanks to Hassan Munir for intellectual and mind-blowing malarkey-related discussions ;) And thanks to everyone else at the Software Engineering Research Group and Department of Computer Science at Lund University for being awesome colleagues and providing a stimulating and fun working-atmosphere.

I would further like to thank Prof. Dr. Daniela Damian at University of Victoria for being a warming host and joyful collaborator. Also, thanks to Sony Mobile in Lund for prosperous collaborations and for making the research relevant for practitioners.

Lastly, I would like to thank those who tolerate me outside of work, like my beautiful and loving fiancee Christine Cleyton Jörgensen for being there every day and standing firm in both ups and downs, and also to the rest of my family and friends for making life joyful, great and awesome in all regards.

Cheers!
Johan Linåker

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 **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.

II **Open Innovation using Open Source Tools: A Case Study at Sony Mobile**
Hussan Munir, Johan Linåker, Krzysztof Wnuk, Per Runeson, Björn Regnell
Submitted to a Software Engineering Journal, 2016 (Under review).

III **How Firms Adapt and Interact in Open Source Ecosystems: Analyzing Stakeholder Influence and Collaboration Patterns**
Johan Linåker, Patrick Rempel, Björn Regnell, Patrick Mäder
22nd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), 2016.

IV **Motivating the Contributions: An Open Innovation Perspective on What to Share as Open Source Software**
Johan Linåker, Hussan Munir, Krzysztof Wnuk, Carl-Eric Mols
Submitted to a Software Engineering Journal, 2016 (Under review).

V **A Stakeholder Analysis Framework for Open Source Software Ecosystems**
Johan Linåker, Björn Regnell, Daniela Damian.
To be submitted.

Related Publications

VI **Requirements Analysis and Management for Benefiting Openness**
Johan Linåker and Krzysztof Wnuk
9th International Workshop on Software Product Management (IWSPM), 2016.

VII **Requirements engineering in open innovation: a research agenda**
Johan Linåker, Björn Regnell, Hussan Munir
1st International Workshop of Open Innovation in Software Engineering (OISE), 2015.

VIII **On infrastructure for facilitation of inner source in small development teams**
Johan Linåker, Maria Krantz, Martin Höst
15th International Conference on Product-Focused Software Process Improvement (PROFES), 2014.

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
Johan Linåker was responsible for writing the paper and and performing the main analysis of the collected survey data. Hussan Munir and Prof. Dr. Per Runeson contributed to the analysis. The survey was planned, designed and distributed by Prof. Dr. Per Runeson, Prof. Dr. Björn Regnell and Claes Schrewelius. The results were reviewed by the all five authors.

Paper II
Johan Linåker and Hussan Munir were responsible for the design, execution, analysis and reporting of the study. Dr. Krzystof Wnuk was involved in performing the interviews. Together with Prof. Dr. Per Runeson and Prof. Dr. Björn Regnell, he was also involved in design and review of the study.

Paper III
Johan Linåker was overall responsible for the study regarding design, execution, analysis and reporting. Patrick Rempel contributed to the design of the study, data collection, time-to-market and innovation-analysis, and reporting of the study. Prof. Dr. Björn Regnell and Prof. Dr. Patrick Mäder contributed by providing reviews throughout the study.

Paper IV
Johan Linåker and Hussan Munir were responsible for the design, execution, analysis and reporting of the study. Dr. Krzysztof Wnuk was also involved throughout the study, including both writing and reviewing the paper. Carl-Eric Mols was involved in the creation the CAP-model and its usage.

Paper V
Johan Linåker was overall responsible for the study regarding design, execution, analysis and reporting. Prof. Dr. Björn Regnell and Prof. Dr. Daniela Damian contributed by providing reviews and design-inputs throughout the study.

CONTENTS

INTRODUCTION

1 Introduction

For software-intensive firms to create a profit and gain a competitive advantage, it is pertinent that their products capture and satisfy the requirements which represent the needs of their customers and target markets [4]. Hence, Requirements Engineering (RE) may be seen as a pivotal practice in adding business value for these firms [47]. Traditional RE describes the need to consider both internal and external stakeholders in the elicitation process [96]. Inside firms operating in a market-driven context specifically, requirements are invented rather than elicited from a specific customer (as in bespoke RE) [142]. This process of inventing requirements is done internally by balancing between a market pull and technology push [85]. Due to factors such as increased globalization, availability of risk capital, and a fluctuating work-force, firms may have to rethink the way this "invention-process" is managed and start to consider how external sources of knowledge could be leveraged [24].

This view is further captured by the innovation management theory of Open Innovation (OI) [24]. The thoery describes how firms should go beyond its own borders in a process of creating, delivering and capturing value (monetary or non-monetary) [24]. Instead of solely relying on their internal research and development, firms are encouraged to open up and take advantage of external ideas, resources and knowledge, e.g., through sourcing or acquisition [34]. In a similar fashion, firms should also consider if and how internal knowledge and intellectual property could be exploited in a more profitable manner externally, e.g., through revealing or selling [34]. Either way, or combined, OI highlights the importance to recognize the external workforce that resides outside of the firm, and the knowledge and competencies that they possess [170].

OI thereby offers a lens through which researchers and practitioners may view and study firms as the focal point and how they interact with an open environment. This has been applied in-depth in a wide variety of industries [27] such as manufacturing [98], pharmaceutical [11], food processing [148], and automotive [79].

However, for software-intensive firms, OI has not gotten as much attention. Munir et al. [127] identify a number of different studies in how OI may be applied, such as inter-firm collaborations [143], crowdsourcing [40], and Open Source Software (OSS) [168]. Out of these examples, OSS was the most studied instance of OI (e.g., [70, 154, 156]). The open environment in this case is constituted by the community of actors that through one or more common incentives surrounds the OSS, and collaboratively see to its development and maintenance [163]. The community may also be framed as a software ecosystem using the definition by Jansen et al. [80] where the OSS constitutes the "common technological platform" that underpins the relationships and interactions between the actors. Through this analogy firms applying OSS from an OI perspective may be seen as members of an OSS ecosystem [114].

To contextualize this relationship from an OI perspective even further, a funnel model may be applied, as originally proposed by Chesbrough [25], see Fig. 1. The funnel represents the focal firm and its internal software development process (1). The funnel is permeable, meaning that the firm can interact with the open environment surrounding it, in our case, an OSS ecosystem (2). These interactions are represented by the arrows going in and out, and can be further characterized as transactions and exchange of knowledge between the firm and the OSS ecosystem (3). Examples of transactions can include software artifacts (e.g., bug fixes, feature implementations, plug-ins, or complete projects), but also opinions, knowledge and support that could regard any step of the internal or external development.

The illustrated interactions 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*). When outside-in and inside-out transactions occur together, the process is termed *coupled innovation* [44]. This may be expected in co-development between a firm and other ecosystem participants in regards to specific functionality.

From a business model perspective, the OSS may be used as a direct part of a firm's product offering, e.g., through an open core or platform-extension model [166], as a basis for support, subscriptions and professional services [26], or as part of a duel-licensing model [170]. However, it may also be the case that the value comes indirectly when the OSS is used as an enabler for the firms' product offerings, e.g., as a development component or as part in the infrastructure supporting the product [69]. It may also be a combination of such direct and indirect factors. E.g., in asymmetric business models, software is made open to instead capture value from additional products, services and data gathering that is managed through the OSS [151]. The potential benefits that motivate these different usages of OSS has its foundation in the external workforce that is available in the OSS ecosystem [127]. As highlighted by OI [24], this can help the firm to strengthen and advance its internal technology capabilities, both in regards to their product and process levels. The new workforce can further help to share the burden of maintenance and development, as well as potentially increase the quality of

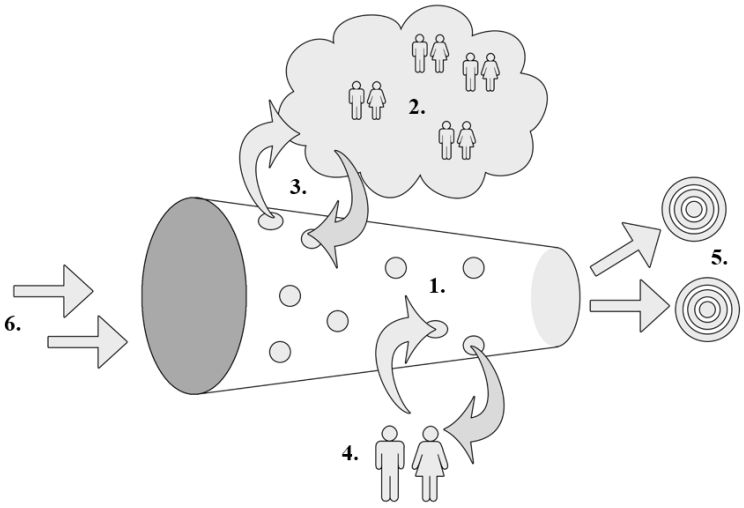


Figure 1: The OI model illustrated with interactions between the firm (funnel) and its external collaborations. Adopted from Chesbrough [25] and paper 4 in this thesis.

the software and decrease in the time-to-market [156].

With respect to RE, an OSS ecosystem membership implies that firms have to consider participation in the external RE process of the ecosystem, and how to bridge such participation with their internal process [107]. In contrast to the latter, the RE process in OSS ecosystems can usually be characterized as being informal and decentralized with a focus on collaboration and transparency [45, 149]. These characteristics further highlight that the focal firm may no longer be the vantage point as the case in the their internal RE process [145]. In the OSS ecosystem the focal firm may be considered as a stakeholder among others in the fluctuating and open stakeholder population. This may imply risks of conflicting agendas [127], difficulty in aligning internal strategies and processes with those of the ecosystem [127], and a need to gain and maintain a suitable position in the ecosystem's governance structure [5] in order to have the influence needed in regards to the ecosystem's RE process [175]. Further, it introduces complexity in regards to what the firm should share with the OSS ecosystem [70]. Giving away differentiating intellectual property, especially to competitors, may have detrimental effects on both for existing and future business [162].

Firms therefore need to consider how they interact with the OSS ecosystems and what software artifacts they choose to contribute, and when. Wnuk et al. [175] describe these activities as central parts of the requirements scoping and management processes. An important output from these processes are what Wnuk et al. refer to as contribution strategies, which is a type of *"management strategy defining when and what to contribute back to the OSS"* [175]. In order to maximize the return on investment (ROI) and influence-building in the OSS ecosystems [175], the contribution strategies should align with how the firm draws value [4] from the OSS projects and their ecosystems [139], e.g., in the form of related business requirements [173]. In extension, the contribution strategies should also consider the relation between the OSS projects, their ecosystems, and the firms' internal product roadmaps and requirements management processes [161].

This thesis aims to explore how firms involved in OSS ecosystems work with the internal and external RE processes, but also to comprehend the challenges implied by the ecosystem membership from an OI perspective. Through this exploration and comprehension, it further aims to create a foundation for a future strategic support that may help firms create contribution strategies and as input, consider and analyze the fluctuating stakeholder population of an OSS ecosystem.

2 Related Work

In this section, related work is presented on RE in OSS ecosystems, its special characteristics compared to traditional RE, as well as characteristics in regards to the governance structures of OSS ecosystems that needs considerations by involved firms.

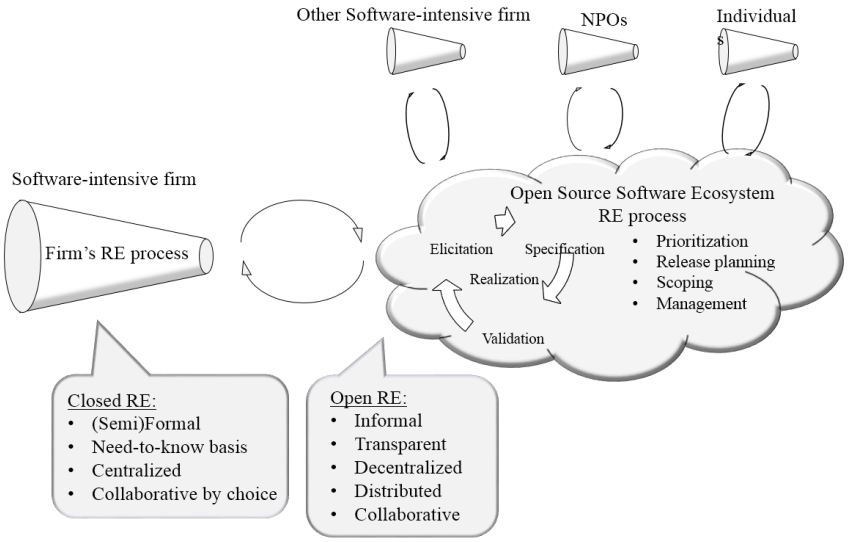


Figure 2: The requirements engineering context influenced by the OI paradigm, contextualized as Closed and Open RE. Adopted from Linåker et al. [110].

In OSS ecosystems, requirements practices are often informal and overlapping [149]. Requirements are commonly asserted through transparent discussions and suggestions by the OSS project's developers and users, often together with prototypes or proof-of-concepts [3, 60]. Assertion may also be done post-hoc, simultaneously as the requirement realization [45, 60]. These assertions are specified and managed in what Scacchi refers to as informalisms [149], e.g. reports in an issue tracker, messages in a mailing list, or commits in a version control system. Through social interaction facilitated by the infrastructure persisting the informalisms, requirements are further enriched and validated [45, 60, 159]. Prioritization is commonly conducted by the core-team overseeing the project management, though care is often taken to the opinions of other developers and users [97]. Ernst & Murphy refer to this lightweight and evolutionary process of requirements refinement as Just-In-Time (JIT) requirements (illustrated by the circular arrows inside the OSS ecosystem in Fig. 1), compared to the more traditional upfront requirements characterized by heavy processes and tool support [45]. Further, Alspaugh & Scacchi contrast how OSS RE steps away from what they refer to as Classical Requirements, characterized as having a central repository, with requirements defined in the problem space, describing the product of need, along with processes for examining the requirements for completeness and consistency [3]. For better consistency, we choose to re-label JIT and OSS RE as Open RE, and

that described as traditional upfront, and classical requirements as Closed RE.

Fig. 2 illustrates the distinction between an internal RE process of the focal firm (represented by the left-side funnel) and an external RE process in the OSS ecosystem (represented by the cloud). As listed, Open RE compared to Closed RE can be seen as being informal to different degrees, e.g., to what level requirements are analyzed and managed [45]. Requirements are often decentralized and distributed over multiple sources, often with a limited tracing. Influence and participation in the work and decision-making are also distributed. Discussions and steering documents are all public and transparent for anyone to see, or participate. Collaboration and negotiation about requirements are key, as consensus often is often needed to make certain decisions [2], although core-team members usually has the final say. Such core-team members and others with a high position in an OSS ecosystems governance structure [5] often attain this influence on the RE process by being active, contributing back, and having a symbiotic relationship with the OSS ecosystem [35]. This governance structure is often referred to as a meritocracy [83]. Nakakoji et al. [128] illustrate this with an Onion model where the outer layer is constituted by the passive user, and the center by the project leader. For each layer towards the center, influence in the ecosystem increases.

As highlighted by Wnuk et al. [175], contribution strategies may be used as a tool for firms to attain such influence. Dahlander & Magnusson [35] describe how a firm can adapt their relationship towards the OSS ecosystem based on how much influence they need, e.g., by openly contributing back to the OSS ecosystem, or keeping changes and new features internal. Based on how open a firm uses the OSS and its ecosystem in their business model and level of influence needed, different strategies may be applied. For example, selective revealing means that differentiating parts are kept internal while commodity parts are contributed [70, 168]. Further, licenses may be used so that the technology can be disclosed under conditions where control is still maintained [168]. In the edge case, everything could be disclosed under open and transparent conditions [26], or even kept closed. As highlighted by Jansen et al. [81], openness of a firm should be considered as a continuum rather than a binary choice between open and closed.

3 Research Goals

To explore and create a foundation for a strategic support on creation of contribution strategies, the following five *Research Goals* (RG) are defined and addressed by this thesis:

RG1: To understand the interrelations between product, process, business and organizational innovation.

RG2: To explore how a software-intensive firm works with Open Source Software ecosystems from an Open Innovation perspective.

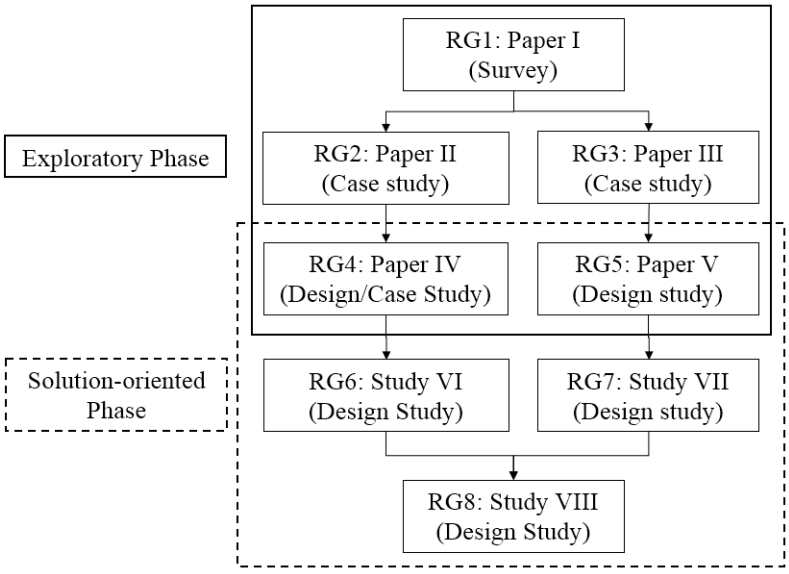


Figure 3: Overview of research goals (*RG1-5*) and related papers (*I-V*) and planned future studies (*RG6-8* and *Studies VI-VIII*).

RG3: To explore how software-intensive firms interact and collaborate with each other in an Open Source Software ecosystem.

RG4: To propose a model that can support software-intensive firms in creating contribution strategies for software artifacts.

RG5: To propose a framework that can help software-intensive firms structure their stakeholder identification and analysis process in Open Source Software ecosystems.

In Fig. 3, an overview is presented to demonstrate how the five research goals (*RG1-5*) are addressed by the five papers (*Paper I-V*) that are part of this thesis. The figure further presents how they relate with future research goals and studies (*RG6-8* and *Studies VI-VIII*) proposed and described in section 8.

RG1 was defined to comprehend different types of potential innovation outcomes of OI and how these different types may be interpreted and interrelated. We adopt the innovation classifications proposed by OECD [1] product, process, business and organizational innovation. To address *RG1*, *Paper I* was designed as a survey [109] performed on the developer organization in a large software-intensive firm.

RG2 was defined to contextualize and understand challenges implied by OI [127], but also to comprehend how software engineering practices are applied when

working with an OSS ecosystem, using the perspective of OI. To address *RG2*, *Paper II* was designed as a case study [147] performed on a development department at Sony Mobile which is involved in the OSS ecosystems of Jenkins[1] and Gerrit[2].

RG3 was defined to understand the temporal dynamics of how software-intensive firms interact and collaborate in OSS ecosystems. To address *RG3*, *Paper III* was designed as a case study [147] that applies social network analysis [165] to explore the stakeholder population of the Apache Hadoop OSS ecosystem[3].

RG4 was defined as a first part of a future strategic support for software-intensive firms to be able to define contribution strategies for software artifacts. To address *RG4*, *Paper IV* was designed as a case study [147] with a design science approach [73] at Sony Mobile to collaboratively develop a prototype model.

RG5 was defined as a second part of a future strategic support with the aim to provide input to creation of contribution strategies for software artifacts. This input is given by helping firms to systematically structure their analysis process of the stakeholder population in an OSS ecosystem. *Paper V* was designed as a design science study [73] and as a continuation of *Paper III*.

4 Research Methodology

A mix of empirical research methods are used to investigate the problem from a real-world perspective [109, 147]. Design science [73] is used to start a transition from an exploratory to a solution-oriented phase (see Fig. 3). Studies were performed both from a focal firm's perspective towards the OSS ecosystem, as well as from an ecosystem perspective with firms considered as stakeholders, to understand the interaction between and the dynamics of an ecosystem' stakeholder population (see Fig. 1). Open Innovation theory is used to frame the problem in to a firm's perspective.

Paper I addresses *RG1* through a survey [109] that was conducted at a large software-intensive firm. The survey was part of an internal project aimed to assess and improve the internal innovation climate of the firm. The questions were a mix of dichotomous, Likert and open-ended, based on literature and executed via an online questionnaire. After an initial pilot, it was sent out via a census sampling to a population of about 900 employees, of which 229 responded (25%). 469 free text answers were generated by the open-ended questions which were analyzed qualitatively through thematic coding.

Paper II addresses *RG2* through a case study [147] at Sony Mobile and its Tools department. Units of analysis were the Jenkins and Gerrit OSS, which are part of the continuous integration tool-chain used by Sony Mobile's product de-

[1] https://jenkins.io/
[2] https://www.gerritcodereview.com/
[3] http://hadoop.apache.org/

velopment, and maintained by the Tools department. From an OI perspective, the Tools department constituted the focal point of Sony Mobile through which the interactions with the open environment (Jenkins and Gerrit OSS ecosystems) was performed. A case-study protocol was created and maintained during the process of conducting the study. The study started with mining and analyzing commit data for both OSS projects, after which a number of sub-projects were identified having involvement from Sony Mobile employees. These sub-projects were then analyzed quantitatively in regards to stakeholder population on a firm-level, and type and distribution of commits. Semi-structured interviews were conducted with three developers at Sony Mobile that were identified through the commit-data. Two more interviews were performed where the interviewees were recommended by the first three. The five interviews were audio-recorded, transcribed and coded thematically [33]. Output from the coding processes were communicated and validated with interviewees.

Paper III addresses *RG3* through a case study [147] of the Apache Hadoop OSS ecosystem. Unit of analysis was the ecosystem's stakeholder population on a firm-level. The stakeholders' interactions were studied based on their contribution of patches to issues included in releases R2.2.0 (2013-10-15) to R2.7.1 (2015-07-06). These issues were mined from the ecosystem's JIRA issue tracker by implementing a crawler. Directed affiliation-networks were created where two stakeholders (represented as nodes) are connected with edges if they had both contributed a patch to a common issue. These edges were weighed based on the size of each stakeholder's contribution relative the other in terms of net changed lines of code. Six networks were created for each second level release (R2.2-R2.7). Developers' organizational affiliation was determined based on email-domain analysis complemented with qualitative heuristical analysis of electronic sources [12, 62]. The six networks, each representing a separate release, were studied in terms of stakeholder collaboration and the influence of the top ten stakeholders based on the three network centrality measures: out-degree, betweeness, and closeness.

Paper IV addresses *RG4* through a case study [147] with a design science approach [73] at Sony Mobile. Iterative design cycles were performed consisting of the steps problem investigation, artifact design and artifact validation. All three steps included informal consultations with four experts at Sony Mobile who are involved in the decision-making regarding the contribution process to OSS ecosystems. Internal documentation of the contribution process and policies was also used and analyzed. A prototype model named the Contribution Acceptance Process (CAP) model was created based on a purchasing and sourcing model proposed by Peter Kraljic [93]. To allow for operationalization of the CAP model, an information meta-model was created to support the communication and follow-up of contribution strategies attached to software artifacts. This meta-model was created, in consultation with experts Sony Mobile and based on an exploratory analysis of Sony Mobile's software artifact repositories connected to the Android platform used in their products.

Paper V addresses *RG5* through a design science oriented study [73] based on *Paper III*. A technology-based artifact is designed in the form of a framework that allows firms to structure their stakeholder analysis process towards OSS ecosystems. The framework is based on three conceptual foundations elicited from literature, and uses social network constructs [165] to enable analysis of stakeholders' interactions and influence on the RE process of an OSS ecosystem. The framework is validated analytically and descriptively [73] through a case study on the Apache Hadoop OSS ecosystem.

5 Results

In this section, results of each paper is presented in terms of addressing research goals as stated in section 3 with the corresponding research methodology as presented in section 4.

5.1 RG1: Paper I

Paper I investigates how product, process, business and organizational innovation [1] interrelates and are perceived in a software engineering context. Findings show that product innovation is perceived to be an enabler for the other three categories, but not the other way around. Some respondents had problems relating to the term, what could be classified as innovative and seeing how they could be innovative in their own roles in the firm.

In the free text answers, several perceived connections and triggers were found between product innovation and the other categories. These can be generalized as new product innovations may require related and interdependent development processes, marketing strategies and organizational structures to adapt and be tailored accordingly. In an opposite manner, new and innovative development processes, marketing strategies and organizational structures may render in new and improved products, e.g., due to new technologies, ideas, resources, faster time-to-market and better quality.

5.2 RG2: Paper II

Paper II investigates how software-intensive firms involved in OSS ecosystems and their software engineering practices in the ecosystems can be contextualized from an OI perspective. The study showed that the main reason for Sony Mobile to "open up" was due to a general shift towards adopting OSS as a platform for the firm's products. As consequence, this mentality transferred to the Tools department and made them adopt OSS for the internal continuous integration tool-chain. Developers were assigned to work with the Jenkins and Gerrit ecosystems in order to tailor and adapt the tools according to Sony Mobile's needs. To build the influence needed to impose their agenda (e.g., in regards to scaling capabilities),

the Tools department's developers were active in contributing software artifacts as well as knowledge and support. They were transparent and communicated their internal tool-chain setup and problems at events and directly with competitors. Main enabler for this openness was that tools and infrastructure such as Jenkins and Gerrit were considered as non-competitive and non-pecuniary.

The Tools department's RE process that interfaced the OSS ecosystems was informal and managed internally in an agile manner through a combination of Scrum and Kanban methods. Prioritization of issues in the OSS ecosystems was made in relation to internal needs. Collaborations were common and often performed on a feature-by-feature basis, including direct competitors. The main outputs of the adoption of Jenkins and Gerrit, along with the collaboration with their ecosystems included new and improved functionality for the two tools, as well as less maintenance and shorter internal release cycles. Main perceived benefit however regarded the flexibility to adapt and tailor the two tools based on internal requirements, rather than troublesome and costly change-requests to customized off-the shelf (COTS) products. Although no metrics were available, it was further perceived that this possibility to tailor the two tools also implied faster build-cycles, higher quality assurance, and in the end improved products with a faster time-to-market. Further, the experiences attained by the Tools-department in terms of working with OSS ecosystems and adaption in development practices has had the effect of introducing an Inner-source initiative to Sony Mobile.

5.3 RG3: Paper III

Paper III investigates how stakeholders' interaction and influence varies temporally in an OSS ecosystem through a case study [147] of the Apache Hadoop OSS ecosystem. The study shows how a previously proposed methodology [135] can be used for stakeholder identification and analysis in an OSS ecosystem and potentially provide inputs to challenges regarding a fluctuating stakeholder population with different agendas [127]. The analysis shows that collaborations occurred between and among competitors and non-competitors. The influence and collaboration fluctuated through releases with complementary views given by the different network centrality measures. A core of stakeholders was present through the releases, while many periphery stakeholders fluctuated with their involvement. Further, a high degree of the issues were isolated (disconnected) which aligns a majority of issues being implemented by the issue-reporters themselves.

5.4 RG4: Paper IV

Paper IV designs and proposes a prototype model to help firms adopt contribution strategies for software artifacts that align with internal product strategies and planning, i.e., if, when, and how they should be contributed to an OSS ecosystem. This synchronization between contribution strategies and internal product strate-

gies and planning allows for a strategic product planning that aligns contribution decisions with a valuation of whether an artifact may be considered as differential or commodity.

From the functional perspective, the Contribution Acceptance Process (CAP) model allows for software artifacts (ranging from bugs to features and larger components) to be classified based on their business impact and control complexity. The former regards how much profit the artifact represents, and the latter how difficult it is to control or acquire the artifact. This classification is done by answering a series of questions and should be done by a cross-functional group of internal stakeholders that can value artifacts according to the two factors. Depending on the classifications, four different types of contribution strategies may be adopted for the artifact. E.g., strategic artifacts are those with a high business impact and control complexity. These are differential and makes up a competitive edge for the firm. These should be developed either internally or in strategic alliances. However, parts that are considered as enablers for the differential functionality such as supporting frameworks may be contributed. A special and rigorous screening process is therefore needed for these artifacts.

To support operationalization of the CAP model, an information meta-model was created. The meta-model presents how a series of software artifact repositories may be setup and linked, from a product platform, via requirements and architectural components, to patches, contributed patches and related commits. This structure allows for contribution strategies attached to a software artifact to be communicated through a development organization, but also to be followed up.

5.5 RG5: Paper V

Paper V designs and proposes a prototype model that allows firms to systematically structure their stakeholder analysis process of OSS ecosystems. Focus is on identifying and analyzing the ecosystem's stakeholders in terms of their interactions and influence on the ecosystem's RE process. Due to the informal and collaborative characteristics of OSS RE [45, 149], and often meritocratic governance structure [83], influence is an important attribute in order for firms to impose their agendas in an OSS ecosystem [35]. Further, as they may be considered as part of a larger set of stakeholders, they have to consider the agendas and level of influence of other stakeholders. The analysis process therefore provides a mean to provide input to how firms in involved in OSS ecosystems should interact and apply their contribution strategies in order to manage conflicting agendas, and build and leverage an influence to align internal strategies and agenda with the ecosystem's.

The framework consists of six steps which considers the nature in how requirements are fragmented as multiple requirement artifacts persisted in a decentralized manner in as many repositories [3, 45]. To create an overview of how stakeholders interact in the ecosystem, the framework describes how the most important repositories should be identified and mined for requirements artifacts based on the scope

and limitations of the analysis. After identification of developers' organizational affiliations, a network should then be created to represent each requirements repository. An influence analysis can then be performed by applying a series of network centrality measures.

6 Synthesis

In this section we synthesize the results from *Paper I-V* in relation to each other and their respective research goals (*RG1-5*) as presented in Fig. 3.

The OI definition motivates its purpose to advance a firm's internal technology capabilities by leveraging the open environment surrounding the firm. *Paper II* illustrates this by observing how firms adopting OSS internally can use the external workforce of an OSS ecosystem as a source of innovation for both Jenkins and Gerrit OSS (*RG2*). However, the outputs should not just be viewed from a product innovation perspective, as some benefits may take the form of process innovations, such as lesser maintenance and faster time-to-market. Further, one type of innovation may also be seen as a trigger for another, either implicitly or explicitly as highlighted by *Paper I* (*RG1*). Product innovation may affect the process with which the products are developed, or the organizational structure containing the processes. This is observed in *Paper II* as product innovations captured in Jenkins and Gerrit spill-of as process innovation with a tailored and optimized continuous integration tool-chain rendering in faster and more stable build-cycles for product developers. A spill-off in terms of organizational innovation is the above mentioned Inner-source initiative.

RE processes in OSS ecosystems may be described as informal and collaborative, which is supported by observations made in *Paper II*. Developers use an agile approach towards the ecosystem prioritizing work based an internal pressing needs. Further, they show the importance of building an influence in an OSS ecosystem in order to affect its decision process. Both offline and online presence is important along with a transparent and open attitude in order to identify and create traction for issues on a firm's agenda. *Paper II* further highlights the presence of co-opetition [129], i.e., collaboration with competitors which can further be identified in *Paper III* on an ecosystem level.

Papers II and IV highlight how software artifacts considered as non-competitive or commodity are shared openly while stricter guidelines apply for those which has a higher business criticality or need of control. Hence, a pertinent aspect when considering what to reveal is how the software artifact and OSS ecosystem in general is used in regards to a firm's business model. In cases where a direct connection applies (e.g., firms with an open-core [166] or asymmetric business model [151]), software artifacts must be viewed in terms of its commoditization life-cycle [162], along with the need to control the spread and development of it. Both these aspects constitute fundamental parts of the CAP model proposed by *Paper IV*. In

cases where an indirect connection exists (e.g., when the OSS is part of infrastructure and build-environments), software artifacts may be more freely contributed back as these are considered as non-competitive and non-pecuniary, as observed by *Paper II*. This is to some extent also captured in the CAP model (see Standardized artifacts) but may need further consideration in following design cycles of the model.

Stakeholder populations in OSS ecosystems can be characterized as constantly evolving with new and unknown stakeholders [107]. There are no roosters of members present due to the informal and decentralized characteristics of OSS ecosystems. This if further observed in *Paper III* which shows how the Apache Hadoop ecosystem has an evolving population where firms' influence and collaboration fluctuates with time (*RG3*). These observations connect to challenges identified in earlier work [127], e.g., difficulty of conflict management, in alignment of internal strategies with the ecosystems', and of building a sustainable influence in the ecosystem. *Papers III and V* raise the importance of continuously identifying existing stakeholders and analyzing their influence on the OSS ecosystems RE process in order to address such challenges. *Paper V* addresses this further by proposing a framework that allows firms to systematically structure their stakeholder analysis process of OSS ecosystems. The framework is focused on analyzing the influence and interactions of the stakeholders, and is contextualized to consider the informal and decentralized nature of OSS RE.

The two topics covered by *Papers IV* and V are tightly entangled. In order to value if a software artifact is differentiating, the firm must first know of if there are any competitors present, some of which may be indirect. Further, presence of potential partners may affect as they may possess complementary know-how or other strategic benefits. To determine the control complexity of a software artifact, the firm must first know how other stakeholders' agendas align or conflict. Hence, the framework proposed in *Paper V* offers valuable input to the decision-support offered by the CAP model presented in *Paper IV*. Combined they offer a foundation for a future strategic support for firms in their requirements scoping and management process towards OSS ecosystems (*RG4-5*), or more specifically, when making decisions about what to reveal, when and to whom in OSS ecosystems.

7 Ethical Aspects and Threats to Validity

We refer to the four aspects of validity as proposed by Runeson et al. [147]: *construct*, *internal* and *external validity*, and *reliability*.

Construct validity refers to what extent the researchers study what they had set out according to their intentions and research questions. In general, existing literature has been an important foundation in all papers to frame and build the research. For example, in *Paper I* and *II* questions are based on literature. In

regards to *Paper III* and *V*, construct validity could be questioned in regards to how issues are generalized to represent requirements. Also this is motivated by how the literature describes OSS RE as decentralized where a requirement can be represented by multiple requirements artifacts (cf. informalisms [149]), spread out over multiple repositories [45].

Internal validity refers to the risk of unknown confounding factors affecting the results. One such risk may concern how network centrality measures [165] are interpreted as an attribute of influence. This interpretation is based on literature (e.g., [46, 131, 145]) and previous empirical studies [135], and further complemented with an analytical validation [73] through a correlation analysis in *Paper V* between the network centrality measures and two positive performance measures for a firm engaged in an OSS ecosystem. No causality should be assumed based on correlation analysis, but indications can be made of an association between the two sets of variables. As the correlation analysis in limited to one OSS ecosystem and for a limited release-set, further analysis validation should be made, also in regards to choice of performance measures.

External validity refers to what extent the findings of the research is generalizable outside the specific case. *Papers I*, *II* and *IV*, these were all conducted at a large software-intensive firm, of which the two latter on Sony Mobile. Settings and conditions may be different at firms of similar size why replicated studies on other case firms could generate different results. However, *Papers I* and *II* were of an exploratory nature to better understand OI in a software engineering context where we believe the results to be generalizable to similar cases. *Paper IV* is a design-oriented study, why further design cycles are needed with applications on other firms with similar and different characteristics compared to Sony Mobile. This also concerns size and maturity of the firm, as startups in various stages may have different needs and issues compared to a large and mature company as Sony Mobile. *Papers III* and *V* were both limited to studying the Apache Hadoop OSS ecosystem. The Apache Hadoop ecosystem was chosen partly because of its high level of firm-affiliated developers, why its stakeholders could be studied on a firm-level which fits the OI context of this thesis. Further, *Paper V*, is based on conceptual foundations which may be considered generalizable from their different perspectives, why the framework is not designed with Apache Hadoop in mind. As with *Paper IV* however, *Paper V* is a design-oriented paper why further design cycles are needed.

Reliability refers to what extent the generation of the research and its findings are dependent on the original researcher. The steps taken in each paper has been documented carefully, and with research protocols used for *Papers I-III*. Triangulation with mixed methods has been used in all papers, along with pilots of survey and interview instruments where applicable. Further, research design and analysis has been discussed and reviewed iteratively among co-authors.

In regards to *ethical aspects*, careful precautions were taken not to reveal sensitive data from the case firms studied by the papers in this thesis. Due to the

empirical nature of the research it is difficult for researchers to avoid coming in contact with this kind of data. It is therefore necessary to abstract the data to a level where the case firm may not feel threatened. However, abstracting too much may render in too vague conclusions and value risk being lost in terms of research contribution. Hence, this is a process of balance. In *Paper I, II* and *IV*, non-disclosure agreements were used, as well as continuous feedback-loops with case firm representatives in order to review the level of sensitivity of what is reported, while maintaining the researchers' integrity and independence. As for *Paper III* and *V*, these are not tied to any specific case firm and all data used is available publicly due to the nature of OSS.

8 Future Work

Future work will be solution-oriented (see Fig. 3) and continue to build on and evaluate the knowledge foundation created through this first and exploratory part, presented in this thesis. Accordingly, the following research goals are defined for future work:

RG6: To continue develop and evaluate the proposed CAP model (Paper IV) through further design cycles to provide a strategic support for software-intensive firms in decisions on what to contribute, when and to whom in OSS ecosystems.

RG7: To continue develop and evaluate the proposed stakeholder analysis framework (Paper V) through further design cycles to help software-intensive firms structure their stakeholder identification and analysis process in OSS ecosystems.

RG8: To integrate results from *RG6* and *RG7* into a complete strategic support for RE in firms involved in OSS ecosystems.

To address *RG6*, *Study VI* is planned to further develop and evaluate the CAP model in collaboration with Sony Mobile and possibly further case firms. The model will be used in workshops with relevant internal stakeholders to create contribution strategies for software artifacts on feature-level. The proposed information meta-model will then be evaluated as a mean to communicate and follow-up the contribution strategies. The study will continue to use a design science approach through iterative design cycles.

To address *RG7*, *Study VII* will continue on the framework proposed in *Paper V* to analytically investigate and validate the attribute of influence on further OSS ecosystems with a high level of corporate stakeholders, such as Linux Kernel[4], OpenStack[5] and WebKit[6]. Analysis results will then be presented to stakeholders

[4] https://www.kernel.org/
[5] https://www.openstack.org/
[6] https://webkit.org/

in the OSS ecosystems through interviews in order to validate the framework. A further aim of following design cycles is to develop a tool that automates tasks suggested by the framework and supports firms in the stakeholder identification and analysis process of OSS ecosystems.

To address *RG8*, *Study VIII* is planned to integrate the technical support-tool and underlying framework from *Study VII*, with the evolved CAP-model from *Study VI* into a strategic support for RE in firms involved with OSS ecosystems. Specifically, the study aims to improve alignment between a firm's requirements scoping [175] towards OSS ecosystems with the firm's internal requirements management [161], product roadmap [92], and business requirements [173], by designing a solution that can help the firm to create contribution strategies in order for its developers to better decide what to contribute and how to prioritize their work. The study will use a design science approach [73, 174] and include application on multiple case firms of different characteristics to strengthen external validity [147].

INCLUDED PAPERS

CHAPTER I

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

Johan Linåker, Hussan Munir, Per Runeson, Björn Regnell, Claes Schrewelius

Abstract

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

1 Introduction

In recent years, the focus on innovation has increased in many lines of business. Novel products and services have always been important, while with an increasing pace of change, new technologies and market concepts being launched, with small vendors coming up and changing the scene in very short time, the need for continuous innovation is stressed in larger companies. Internet technologies for communication and distribution, and products and services primarily differentiated with respect to software, enables this shift by lowering the thresholds for new actors, and thereby threatening the position of existing ones.

Innovation is not only bringing new products to the market. The Organisation for Economic Co-operation and Development (OECD) Oslo manual [1], which is used to guide national statistics collection on innovation, distinguishes between four categories of innovation, i) product, ii) process, iii) marketing, and iv) organizational. These categories are defined as follows: *A product innovation is the introduction of a good or service that is new or significantly improved with respect to its characteristics or intended uses* [1, §156], while a *process innovation is the implementation of a new or significantly improved production or delivery method* [1, §163]. In the context of software engineering, we also count software development processes and practices as “production" methods in the process innovation category. *A marketing innovation is the implementation of a new marketing method involving significant changes in product design or packaging, product placement, product promotion or pricing* [1, §169]. Note that this involves the whole concept of bringing a product or service to the market, a kind of innovation we have seen in the software and internet domain, for example, using information or advertising instead of money as a trade for services. Finally, an *organisational innovation is the implementation of a new organisational method in the firm's business practices, workplace organisation or external relations* [1, §177]. This is also prevalent in software, where for example open source software, outsourcing and offshoring significantly has changed the game in many lines of business.

Given these categories of innovation, we were interested in studying to what extent these were known and integrated in the culture of a large company, which is under rapid change, and where innovation is a key survival factor, due to the volatility of the market. In particular, we wanted to study the awareness of the innovation concepts, and the interplay between the four types of innovation; which types precedes the other? There is a similarity to the software process improvement trinity of people, process and technology, much discussed in the 1990's [77]. More specifically, this study formulates three research question:

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

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

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

To address the research questions we launched an internal online survey [50] in a local branch of a multi-national corporation. The target population consisted of approximately 900 employees. On a global level the company employs approximately 5,000.

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

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

2 Related work

Innovation related to information technology (IT) has become vital part of most organizations' success, primarily for two reasons: i) growing importance of innovation for organizational life, and ii) the introduction of IT into almost every business unit of organizations [49]. Lee and Xia [99] addressed the process bottlenecks to innovation, where development teams are inefficient and reactive in most cases. Consequently, this causes problems with lack of support for business adaptions to shifting demands. Agile development seem to offer remedy to make the whole process more innovative for product development and help development teams to quickly deliver innovative, high quality solutions to an ever increasing demand of business innovation [74].

On the other hand, research evidence [30] also suggest that agile could also be a hindrance for product innovation. It creates barrier in transferring the ideas outside the team boundaries due to short iterations and feature backlog reduced the amount of time that teams could spent trying new things or sharing new ideas across different teams. Wnuk et al. [175] also hinted the fact that existing requirements processes are designed to handle mature features and consequently, raises the question of process innovation by having a separate requirements engineering process to make room for innovative features (other than featured backlog) in the products.

Lund at al. [112] conducted a survey to explore the effects that reutilization have on innovation. Results revealed that standardization of process will free up time for innovation and most interestingly, routines are capable of having positive impact on occurrence of ideas and follow through on ideas. Furthermore, paring routines with openness to continuously improve the existing routines leverage positive effects on innovation. Therefore, take away from the study for managers is to

take a look at existing routines with the spectacle of improving them, which will not only improve the efficiency but also the innovation aspect.

Moreover, another study was found where Harrison et al. [66] conducted a survey with 170 Finnish software organizations to explore the impact of human capital on open innovation. Therefore, it can be used as an example where people are affecting the innovation activities in the organization. The study findings suggest that software companies with the larger academically educated staff are more likely to apply open innovation business strategies to accelerate their internal innovation process. The study further argued that this could be due the strong ties between communities and universities. Similarly, Nirjar [132] also performed a survey with 121 software companies across India to explore the impact of workforce commitment on the innovation capability of the software enterprises. The study findings highlighted that the commitment of the managers of software firms can significantly enhance the innovation productivity by creating certain policies (i.e. open business model) [24] and practices/processes.

3 Methodology

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

3.1 About the company

The company, which is a multi-national corporation with approximately 5,000 employees globally, develop embedded devices and the studied branch is focused on software development for communication hubs and additional connected devices in an internet of things (IoT) fashion. We consider the studied company a representative case [147] for similar ones, and hypothesize that the findings have a much broader generality than just this company. The studied branch of the company has 1,600 employees, of which 800 work on software development for the devices, and 100 work on connected devices.

The company develops software in an agile fashion and uses software product line management (SPL) [138]. The company has defined more than 20,000 features and system requirements across all the product lines. Considering the innovation aspect, the company is moving from a closed innovation model to an open innovation model [24], through the use of open source software to exploit the external resources to accelerate their innovation process. The open source solution, referred to as *the platform*, is the base for their software product line projects and derived products. New projects on the product line typically entails 60 to 80 new features with an average of 12 new system requirements per feature. There are more than 20 to 25 development teams develop these features.

3.2 Survey design

An internal online survey [50] was designed in collaboration between the researchers and company representatives, running an internal project, aimed at assessing and improving the innovation climate in the company. The questionnaire is composed of three major parts:

1. Factors that contribute to the innovation climate, based on Ekvall's scheme [43].
2. Questions on the four types of innovation (product, process, organizational and marketing) and their relation, based on the OECD model [1].
3. Factors that hinder and help innovation, based on Jansen et al.'s Open Software Enterprise model [81].

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

One particular term was given certain care, namely *marketing innovation*. The original definition is that a *marketing innovation is the implementation of a new marketing method involving significant changes in product design or packaging, product placement, product promotion or pricing* [1, §169]. However, in the company context, the term was perceived to be only related to what the marketing department was responsible for, and thus too narrow. Therefore, we replaced the term with *business innovation* and extended it to cover the process where the needs of the customers are captured as input for the product planning. This extends business innovation into the area of Requirements Engineering, which can be seen as a software engineering process, i.e. is covered by the process innovation definition. This area is therefore somewhat overlapped, but with the general distinction that high level capturing of requirements is mainly covered by the business innovation definition.

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

3.3 Survey analysis

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

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

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

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

Qualitative analysis with a thematic approach [33] was used to analyze the data, which was codified in up to three levels. Based on the codified data and the comments in general, perception of innovation concepts were analyzed (Subsection 4.1) and the connections between product innovation and process, business and organizational innovation, respectively were identified (Subsections 4.2–4.4). Further on, based on the themes and comments in general, challenges were then identified and generalized in regards to the four types of innovations (Subsections 4.5–4.8).

3.4 Threats to validity

The construct validity [86], refers to whether the survey measured what it was intended to. This can be addressed through e.g. pilot studies, which was performed before the official launch. Further on, the questions were developed iteratively and based on established literature.

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

External validity regards whether the results be generalized to outside of the surveyed sample [147]. In this paper, we analyze the questions, which can be published from the company's confidentiality perspective. Thus, we do not focus on their perceived current innovation status, but rather on the general understanding of innovation factors and their relations. Thereby, we also focus on the most generalizable aspects, which we hypothesize are valid for other companies of similar characteristic to the studied one, as a representative case [147].

A surveys reliability [86] concerns whether the same results can be obtained if the survey process was repeated. As the sample was obtained through a census sampling frame and had a response rate of 25% we regard this optimistically. Although, this cannot be strengthened until follow-up surveys are performed. This is something that will be done in the future as the company wants to measure how the internal perception of innovation develops over time.

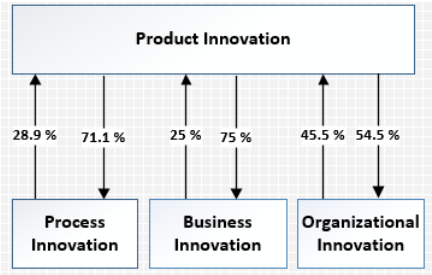


Figure 1: Triggering relation between the four types of innovation: product, process, business and organizational. Percentage value shows the share of respondents that select X → Y as the most likely scenario.

4 Results

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

4.1 Perceptions of innovation

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

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

A reason could be unawareness of what the company counts as innovations and examples of different types of innovations. *"I don't know much about the innovations that we do. I didn't know about the [example feature] for instance"*.

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

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

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

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

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

4.2 Product innovation vs Process innovation

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

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

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

Another challenge is idea tracing and execution uncertainty (see Subsection 4.5), which is an area where we hypothesize that well-designed processes can help to clarify what happens to ideas and the roadmap for how innovations can be pushed

through. Similarly, processes can also help to increase the awareness of the product scope and the innovation strategies in the organization.

Process innovation may help the organization become more efficient and reduce waste as can be interpreted by the OECD definition [1] and as pointed out by a respondent: *"... process innovation improve performance, simplifies and speeds-up development process - thus allowing to have more resources in true product innovation"*. This aligns with the area of Software Process Improvement [67], which includes possible implications from new or improved tools and techniques. As put by another respondent: *"... We need to have the proper techniques, equipment and SW in order to develop new and improved products."*

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

Beneficial factors from a process change, other than freed up resources, may also include an increase in performance and quality as confirmed by the respondents. Although, it is a matter of definition how software quality relate to product innovation [140], this will hopefully render in a better product offering which further down the release ladder may prove to be a trigger of future product innovations.

Hence, by innovating and improving the processes in the correct way and dedicating the freed up resources to product innovation, process innovation can be seen as a trigger for product innovation. This is in line with findings by Lund and Magnusson [112]. On the other hand, processes are not decoupled from the products. There needs to be an awareness of product roadmaps and an adaptive mindset as some processes may require continuous tailoring as a consequence. *"I think the general mindset is "keeping the eye on the prize", you see the upcoming releases in the horizon and you adjust the process to meet those releases."*

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

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

4.3 Product innovation vs Business innovation

On the question whether product innovation triggers business innovation, or the other way around, 75 percent answered the former (see fig.1). As with the previous

question, although there is a clear majority in one direction, this does not give the complete answer.

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

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

New innovative ways are continuously needed to keep pace and capture the demands from the existing and emerging customer channels, e.g. through end-user feedback [6]. An awareness of what needs the customers have today and will have tomorrow, is an important input from business and marketing to push the product innovations forward in the right directions. *"Because business innovation brings in new experience directly from market, new demands and requirements and thus giving a product a right direction"*

This creates a challenge for the organization in terms of synchronization. The view of what features are to be considered game-changers and prioritized in the release planning process [23], may prove troublesome due to internal communication gaps between marketing and product development [84], which may lead to wrong features being promoted as a consequence. *"Scope/product planning, business side and development [should be] in sync regarding both our innovation initiative [. . .] and how to drive innovations all the way to product."*

As explained, there is a dual sided relationship. There is a dependency going in both directions where one can trigger the other. One respondent provided a concrete example which summarizes the relationship. *"It is pretty much both. Look at the music and film business which has invented new ways of marketing*

and distribution, but I believe the wish of distribute TV via satellite has created new products for making it possible and to get paid for it. Then again we have the Google glasses. Right now they are cool, but not very useful until we find a useful feature for them and that itself will create a business for them."

4.4 Product innovation vs Organizational innovation

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

Improving and innovating the way in which a company collaborates and interacts with external parties and stakeholder, can trigger product innovations in several ways. Application of open innovation business strategies is one way to accelerate their internal innovation process [66]. Crowdsourcing ideas, engaging in Open Source communities, welcoming third-party developers, acquiring promising startups and starting joint-ventures or ecosystems are a couple of activities that falls into the open innovation paradigm originally defined by Chesbrough [24], that may render in new product innovations.

Creating a more innovative organizational environment with committed employees is another way that can lead to more product innovations [132], as described by a respondent: *"With a flexible and happy organization that makes people get looser boundaries I believe we can get a more innovative climate"* Bringing people from different backgrounds and functional areas creates diversity and enables for new discussion to arise and to discuss ideas from new angles [21, 90], or as put by the following respondent: *"Connecting colleagues which hadn't possibility to communicate before allows to discuss more problems and ideas."*. Calantone et al. [21] adds that this cross-functional integration also allows for the employees to evolve their skills by learning and sharing knowledge amongst each other, which is important for product development.

This connects to a need for a general awareness of what has been done, and what is being worked on. *"... more often than not these innovations are "hidden" in small segments of the company, not actively promoted and spread (and that's both good and bad, many projects dies when they need to become too big)."* By communicating items such as features, functionality, experienced problems and related solution across internal borders, cross-functional views can be established more automatically. A solution in one project may turn out to solve the same issue or create new ideas in another project, which could either be considered a process or a product innovation. This relates to the concept of inner source [105] and how it can help organizations work more open and cross-functional, and in the end become more innovative [122].

Organizational barriers and communication issues is another area, where organizational innovation may trigger product innovation in the long term perspective.

When products or processes stretch over multiple business units or projects, this can create room for bureaucracy, different prioritization schemes, culture and politics, to mention a few factors [90]. *"Some sections within the company are quite innovative, but when it comes to cross-functional agreements and alignment, there always seems to be a resistance to change and adapt to new ways of working and safeguarding what seems to the best for "me/my team" is more important than what's best for the company."*

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

Looking from the other perspective, new product innovations will create new demands and implications which will give rise for possibilities and triggers for organizational innovation [21]. *"New and exciting products means we have to adapt how we work to support these in the best-possible, not only from an engineering or software perspective, but for example from the launch projects etc."*

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

4.5 Product innovation challenges

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

a) *Idea tracing and execution uncertainty* – Even though there may be a rich pool of innovative ideas being produced and a general will to contribute, it is important to maintain and support it. Knowledge and awareness of what happens to ideas contributed to the innovation development process is important for the contributors to feel that they are taken seriously and that it is worth to continue contributing, which in turn gives an increased innovation capacity for the company [90]. When the ideas come bottom-up there needs to be a feedback loop top-down that stimulates this need of information as confirmed by Koc and Ceylan [91], and Wnuk et al. [175].

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

c) *Product scope and innovation strategy* – Uncertainty about the product roadmap and feature scope leads to risks that the creative minds of the company are misdirected. A common and established innovation strategy can help defining the product scope and frame where ideas are needed suggested by Koc and Ceylan [91], and Wnuk et al. [175].

d) *Limiting environment and mindset* – Soft factors such as employees feeling that they can have a free mindset and share ideas openly is important for an innovative environment. It must be okay to test new ideas, but also to fail. These are factors, triggered by Ekvall's innovation climate model [43].

e) *Restriction by external stakeholders* – A commercial product company can have many stakeholders, some not being the end customer. This may include distributors and service providers further down the value chain, adding value and modifications to the product before they reach the final buyers. These stakeholders put requirements that may prevent and limit the feature scope possible to address. This filter risks to kill ideas inside the company and ignore needs, both identified and unidentified, from the end customers. This challenge is in line with Conboy and Morgan's findings [30].

f) *Limited time for innovation activities* – Tight project budgets and short deadlines are two factors that can restrict time available for idea creation. Developers usually have pet projects and ideas they would like to work on, some even dedicate their spare time for this purpose. By allowing the time, this can prove a valuable source of product innovation as suggested by Conboy and Morgan [30].

g) *Cross-functional resources* – Bringing new people together creates new product ideas and can boost innovation development. Cross-functional labs-sections and dedicated innovation team are two examples suggested by Conboy and Morgan [30], and Koc [90].

4.6 Process innovation challenges

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

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

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

c) *Process change top down* – Problems can arise when a process is introduced top-down instead of bottom-up. Managers may not always know what is the most efficient way to work compared to those actually performing the work. This challenge is also in line with the findings of Qin [141], and Wnuk et al. [175].

4.7 Business innovation challenges

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

a) *Reaching the end-customers* – When there are layers between the producer and end-customer, for example, distributors and service providers, promotion of new ideas and product innovations to end-customers gets complicated. As technology and social habits evolve, new innovative ways are needed to keep pace with the different forums for communication used by the end-customers of today and tomorrow. Examples of such phenomena are software ecosystems [176].

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

4.8 Organizational innovation challenges

Organizational innovation challenges relate to collaboration, communication and change.

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

b) *Intra organizational collaboration* – Barriers and layers can prevent otherwise prosperous and potential collaborations between business units in organizations. Examples may be different sub-priorities of features between projects and multiple number of mangers creating a complex and bureaucratic hierarchy as identified among the respondents and confirmed by Koc [90]. These are related to what Bjarnason et al refer to as “gaps” [15]. Koc further points out that such cross-functional integration demands a high level of coordination, otherwise it will rather have a negative impact on the product innovation.

c) *Intra organizational learning* – Unawareness of what has been done in other parts of the company can create inefficiency and missed possibilities. In regards to process innovation, tools, technologies and processes from one part may prove

its self superior or complementary to those used in other parts. And in regards to product innovation, a commoditized good or service from one business unit may turn out as innovative if added to the value proposition in another business unit's product chain. This is a challenge in-common with inner source [105], but also one of the ways in how it can help organizations become more innovative by using it as a type of intra-organizational open innovation [122].

5 Conclusions

The view on what innovation is and where it can be performed is a diversified topic. OECD [1] differentiates between four types: product, process, market and organizational innovation. These were adopted in the survey on which this paper is based on, with a redefinition of market innovation into business innovation. The original definitions are general and applicable on a multiple number of fields. This paper puts them in the context of software engineering characterized by the opinions of people involved in different levels of a large software development organization.

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

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

Challenges correlated to the different innovation types were also identified, with respect to the third research question (See **RQ3**, Section 1). These give a context to the term of innovation that covers parts other than the more normal

conception of innovation in regards to just products. Some challenges may target more than one type of innovation, e.g. internal communication which can cause issues for introduction on new processes and organizations as well as hinder ideas to be spread and discussed.

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

CHAPTER II

OPEN INNOVATION USING OPEN SOURCE TOOLS: A CASE STUDY AT SONY MOBILE

Hussan Munir, Johan Linåker, Krzysztof Wnuk, Per Runeson, Björn Regnell

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 realization 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 company (Sony Mobile) in OSS communities from an OI perspective and highlights the innovative outcomes resulting from OI. We have also explored what SE practices that has been adapted in relation to OI. **Method.** An exploratory embedded case study design is used to investigate how Sony Mobile use and contribute to Jenkins and Gerrit; the two central OSS tools in their continuous integration tool chain. Quantitative analysis is performed by extracting the change log data from source code repositories in order to identify the top contributors 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 company. ii) Assets which are not competitive advantage nor a source of revenue are left open, and gradually, the company turns more and more open. iii) The requirements engineering process towards the community is informal and based on engagement. iv) The need for

systematic and automated testing is still in its infancy, but the needs are identified. v) The innovation outcomes include free features and maintenance, but also increased speed and quality in development. **Conclusion.** Moving from Closed Innovation model to Open Innovation model was a paradigm shift 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. Furthermore, the company only choose to open up all those the projects to the communities that are not the main source of revenue. Future work includes investigation of other contexts for which the presented findings may be relevant.

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 [24] explains this through the term *Open Innovation* (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"*. To further explain the creation of this new paradigm, Chesbrough refers to *erosion factors* that undercuts the logic of the *Closed Innovation* model of R&D and provides the logic behind the OI model. These erosion factors include increased mobility of workers, declining US hegemony, more capable universities, and a growing access of venture capital to start-up firms. These factors have changed the conditions under which firms innovate. In addition, the rise of Internet has brought knowledge access and sharing capabilities of previously firm-specific internal networks to the World Wide Web [28].

For software organizations, Open Source Software (OSS) provides a common example of OI, and how external resources and knowledge may be leveraged internally to provide benefits such as an increased innovation, quality and shorter time-to-market. Likewise, internal software may be released externally to new or existing OSS communities in order to profit from the same benefits. The increased openness also poses significant challenges to software organizations in terms of securing their competitive advantage [127]. One such challenge may be to manage the increased number of both known and unknown stakeholders. This complicates the governance of the OSS project with more opinions, which in turn affects what requirements that should be selected, when these should be released, but also the overall road-map for the OSS project [108, 175]. Other recognized challenges regard uncertainty of what to contribute, when, and how to maintain a differentiation towards competitors that may also be involved in the OSS community, or users of the OSS project [70, 81, 162].

To manage these challenges, firms engaged in OSS communities need to adapt and innovate their internal software development strategies and processes. For ex-

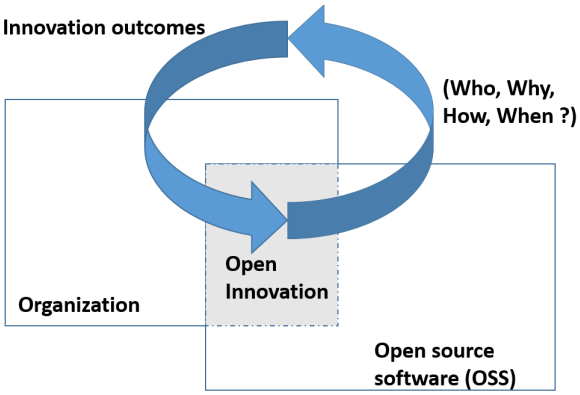


Figure 1: Study Objectives in the intersection between proprietary organizations and open source software.

ample, better influence on feature selection and road-mapping may generally be gained through a more active participation, as many OSS communities are based on meritocracy principles [83]. Also, some benefits may first be fully utilized after contributing back certain parts to the OSS community [163]. For example, by correcting bugs, actively participating in discussions and contributing new features, a firm might reduce maintenance cost compared to commercial software development [156]. Hence, in order for a firm to gain the expected benefits of OI in regards to their products, internal process innovations may be a required step on the way forward [95, 144, 175]. Existing literature does not go into detail on how these internal SE process adaptions should be structured or executed [127]. 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 adopt OI through the use of OSS, and what innovative outcomes that follow as a consequence (see Fig. 1). 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 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 [106] 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 followed. Section 2 highlights the related work and

Section 3 states the research methodology. In Sections 4 and 5 results from the quantitative and qualitative analysis are presented, respectively. Finally, Section 5 explicates a discussion in regards to the results followed by the conclusion in Section 7.

2 Related Work

Below we provide relevant background needed to put this study into the contexts of OI and OSS. The related work in this study is partly based on the systematic mapping study by Munir et al. [127].

2.1 Open Innovation and Open Source Software

OI is a model that was defined in 2003 by Chesbrough [24]. It was based on his observations at Xerox PARC and how they used external knowledge inside of their R&D organization, as well as internal knowledge outside, to advance their technology and innovation. These different directions of knowledge flow are referred to as *outside-in* and *inside-out* processes respectively [24]. These processes consider events and actions of a one-time character, e.g., selling of an IP (inside-out) or acquisition and integration of a start-up (outside-in). In cases where the focus is on co-creation where firms must both give and take (e.g., alliances and joint-ventures), the process is referred as a *coupled process* [44].

The OI model is general in the sense that it can be applied in many contexts. In Software Engineering (SE), a common example is the use of OSS [127], which as a phenomena existed before the OI model was defined [101]. The model sets focus on the single firm and how its use of OSS can help improve its technology and innovation. The innovation may have a direct or indirect impact on the company's product or process innovation. Product innovation refers to a good or service that is new or a significantly improved product with respect to its characteristics or intended use, and process innovation is an implementation of a new or significantly improved production or delivery method [106].

IBM's engagement in the Linux community exemplifies how a firm can leverage OSS from an OI perspective. IBM has donated hundreds of patents and invested more than $100 million a year to support the Linux OS. Consequently, risks and costs of development were shared among other stakeholders such as Intel, Nokia, and Hitachi, which also has made significant investments in the Linux community [100]. The idea behind IBM's investment in Linux was to 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 customer [28].

2.2 Open Innovation Strategies

Dahlander & Magnusson [36] show how firms may use the OSS communities in regards to their business model from an OI perspective. A firm may access the communities in order to extend the firms resource base, align the firm strategy with that of the OSS community and/or assimilate the community in order to integrate and share results with them. Dahlander & Magnusson further explain in another study [35] how the relationships between the firm and the OSS community may be of different characters, e.g., by symbiotically giving back result to the community, or as a free-rider keeping modifications and new functionality to oneself. Pending on how open a firm chooses to be in regards to their business model, different strategies may be enforced. By selectively revealing, differentiating parts are kept internal while commodity parts are made open [70, 168]. With licensing schemas (cf. Dual-licensing [24]), technology may be fully disclosed but under such conditions that competition may not exploit the OSS that may hurt the focal firm [168]. Alternatively, everything may be disclosed under open and transparent conditions [24].

In the case where firms choose to selectively reveal, they need to be able to systematically judge what parts are to be considered commodity or of differentiating value. This is made further complicated as functionality sooner or later will be become commoditized as a result of a constantly progressing technology life cycles [162]. To consider these aspects, firms should construct a contribution strategy that can help them to focus their internal resources on value creating activities, rather than contributing unnecessary patches or differentiating features [175]. Pending on what benefits or strategic goals a firm wish to achieve, different rationale may exist for what is to be revealed. Based on a case study of the Linux kernel community, Henkel [70] reports how small firms 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 [72] further reported how openness had become a competitive edge, as customers had started request even more revealing.

Dahlander & Wallin [37] shows how having an employee in the community can be an enabler for the firms to not only gain a good reputation but also to influence the direction of the development towards the firms' own interests. However, to gain the roles needed, individuals need to contribute and become an active part of the communities as these are often based on the principles of meritocracy [83]. Krogh at al. [164] investigated the strategies and processes by which newcomers join the existing communities, and how they initially contribute code. The study developed a construct entitled *joining script*, and proposed that Committers who follow joining scripts (offer bug fixes, report bugs, take part in discussions, give feedback etc.) are more likely to obtain access to the community. Consequently, a developer is granted access to a privileged source code commit regime.

2.3 Challenges of Open Innovation in Software Engineering

Stuermer et al. [156] conducted a study on implementing a private collective model at Nokia to identify the incentives for firms and individuals in investing OSS. The study examined the development of the Nokia Internet Tablet which builds on a hybrid of OSS and proprietary software development. The results indicated that the cost of the model are high, in terms of difficulty to differentiate, guarding business secrets, reducing the community barriers and giving up organizational control. On the other hand, Nokia reaped benefits in terms of low knowledge protection costs, learning effects, reputation gain, reduces development and innovation costs.

Wnuk et al. [175] studied a global software producing firm that uses an OSS platform in its embedded hardware products. The study identified a series of challenges in regards to the firm's requirements scoping and management processes. The study highlights the need for a contribution strategy, explaining what and when to contribute back to a community. Not contributing back was perceived as a risk at it can create unnecessary maintenance and gap analysis. Internal prioritization and release planning of requirements were also identified as problematic areas as the firm needs to consider the corresponding activities in the OSS community

West et al. [171] examined the complex ecosystem surrounding Symbian Ltd. and identified three inherent difficulties for firms leading an OI ecosystem: 1) prioritizing the conflicting needs of heterogeneous ecosystem participants, 2) knowing the ecosystem requirements for a product that has yet to be created, and 3) balancing the interests of those participants against those of the ecosystem leader.

Van der Linden et al. [162] argued that software products lose value with the passage of time due to ever growing and improving software components, and thereby products become a good candidate for OSS development.

Daniel et al. [39] examined how conflicts between OSS communities and firms affect the organizational commitment of developers towards their organizations. The study suggested that the conflict between organizational and OSS standards reduces developers' organizational commitment. However, it is strongly dependent on the degree to which developers associate themselves with organizations or OSS communities.

2.4 Open Source Development Practices inside an Organization

Mockus et al. [119] suggested that commercial and OSS practices might be fruitfully hybridized in a number of ways, commonly referred to as Inner Source [155]. For instance, it might prove attractive for commercial developers to use the OSS style project structure where a core team of recognized experts who alone have the power to commit code to an official release, and a much larger group who con-

tribute voluntarily in many ways. Furthermore, findings suggested that there are two areas that seem particularly promising for the introduction of OSS techniques in the commercial world namely platforms and tools. Firstly, distinct products can be built on top of a platform, saving a shared common basis. Secondly, developers create a variety of in-house tools to facilitate their own work. Since users are the developers of the tools therefore, itâĂŹs a relatively safe place for trying out OSS techniques with low risk. The focus of this study revolves around the tools entitled Jenkins and Gerrit.

2.5 Summary

Research has shown a lot of interest for the OI and its different applications [168], including how firms leverage OSS from an OI perspective [127]. However, focus has been rather limited to management and strategic areas (e.g., [36, 156, 169]), with some exception of inner sourcing [121, 155]. Little is still known about what triggers software organizations to adopt OSS from an OI perspective and how this affects SE practices [127]. This study contributes by studying why and how a software organization opens up and the use of OSS from an OI perspective for creating internal product and process innovation [106], and how their SE practices are adapted.

3 Case Study Design

Below we describe the research design of this study. We explain chosen research questions, structure of the case study design, and methodologies used for data collection as well as for our 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 from a closed state, and how this impacts on the organization's 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 [133] and Gerrit [76].

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*** is an OSS code review tool created by Google in connection with Android in 2007. It is tightly integrated with the software configuration

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 Mobiles' 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 the Sony Mobile should contribute functionality.
RQ4: What are the innovation outcomes as a result of OI participation?	To explore the vested interest of Sony Mobile as they moved from a closed innovation model to an OI model
RQ5: How do the requirements engineering and testing processes interplay with 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.

management tool GIT, working as a gatekeeper, i.e. a commit needs to be reviewed and verified before its allowed to be merged into the main branch.

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 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 [106] 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.

Based on this background, and the research gap identified in earlier work [127], we formulate our 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

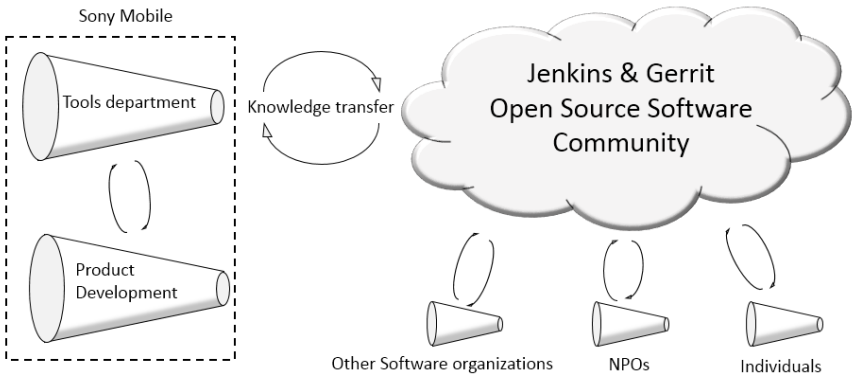


Figure 2: The Jenkins and Gerrit OSS communities surrounded by Sony Mobile and other members. Arrows represent knowledge transfer in and out of the community members, which in turn are illustrated by funnels, commonly used in OI literature [24].

fixes, new features, documentation etc.). *RQ2* and *RQ3* explore the rationale behind Sony Mobile's transition from closed innovation to OI. Furthermore, *RQ4* highlights the key innovation outcomes realized as a result of the openness. The final research question (*RQ5*) 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 the quantitative analysis of repository data, while the remaining four research questions (*RQ2, RQ3, RQ4, RQ5*) are investigated 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 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 in an agile fashion and uses software product line management with a database of more than 20,000 features suggested or implemented across all product lines [138].

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 the active involvement and community contribution with many code commits to Jenkins and Gerrit. This maturity results in a transition from closed

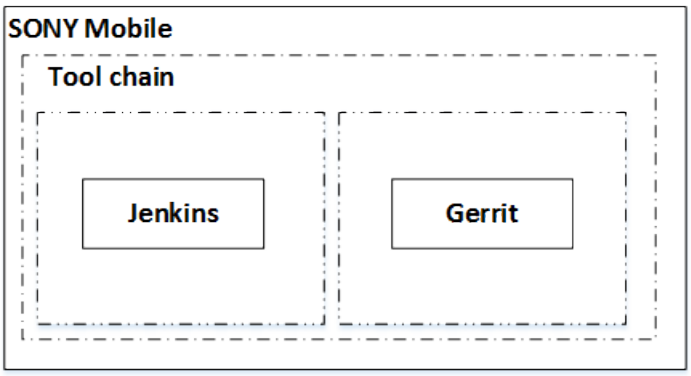


Figure 3: Units of Analysis

innovation to OI [24], given that business values are created or captured as an effect.

From an OI perspective, there are interactions between the Tools department and the Jenkins and Gerrit communities (see Fig. 2). The in- and outgoing transactions, visualized by the arrows, are data and information flows, e.g. ideas, support and commits, can be termed as a coupled innovation process [44]. 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 Jenkins and Gerrit communities into the product development teams of Sony Mobile. This creates another set of transactions inside Sony Mobile which can be labeled as process innovation [1] in the sense that new tools and ways of working improve process 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. [25]. Therefore, Sony Mobile is chosen as a suitable case for OI in SE.

We applied case study design with Jenkins and Gerrit as units of analysis [147] as these are the products in which the exchange of data and information enables further innovation inside Sony Mobile, see Fig. 3.

3.3 Case Study Procedure

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

1. Preliminary investigate Jenkins and Gerrit repositories

2. Mine the identified project repositories.

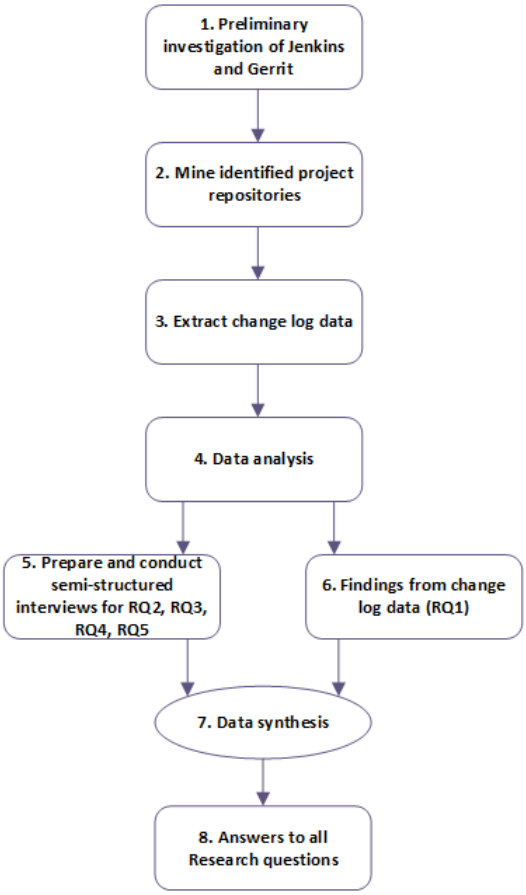


Figure 4: Case Study Procedure

3. Extract the change log data from the source code repositories.
4. Analyze the change log data (i.e. stakeholders, commits etc).
5. Prepare and conduct semi-structured interviews .
6. Analyze the qualitative data to answer *RQ2–RQ5*.
7. Summarize the findings from the change log data to answer *RQ1*.
8. Synthesize data
9. Answer the research questions *RQ1–RQ5*.

3.4 Methods for Quantitative Analysis

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

Preliminary Investigate Jenkins and Gerrit Commits

A commit is a snapshot of the developer's files after reaching a code base state that needs to be remembered. The number of lines of code in the commit may vary depending upon the nature of the commit (e.g. new implementation, update etc.) [68]. The comment of a commit refers to a textual message usually related to the activity that generates the new piece of code or an updated committed code. It ranges from a simple note to a detailed description, depending on the project's conventions. In this study, we used the keywords in Table 2 as a reference point to classify the commit messages [68]. Contribution is a collection of commits that are sent to the community.

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 [117]. 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. Committers to the communities, 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 emails 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 was used, as suggested by earlier research [13,62]. 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 limited evidence of commits to the Jenkins community was identified. The reason for this was that Jenkins is a plug-in-based community, i.e. there is a core component 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]
- Jenkins-Gerrit trigger (Jenkins plug-in)[4]
- Build-Failure-Analyzer (Jenkins plug-in)[5]
- External resource viewer (Jenkins plug-in)[6]
- 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. [68]. 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 or the code without adding new features.

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

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

Multiple researchers were involved in the commit message classification process. After defining the classification categories, Kappa analysis was performed to calculate the inter-rater agreement level. First, a random sample of 34% of the total commit messages were taken to classify the commit messages and Kappa was

[1] https://github.com/jenkinsci/gerrit-trigger-plugin
[2] https://github.com/sonyxperiadev/pygerrit
[3] https://gerrit.googlecode.com/svn/documentation/2.1.2/cmd-stream-events.html
[4] https://wiki.jenkins-ci.org/display/JENKINS/Gerrit+Trigger
[5] https://wiki.jenkins-ci.org/display/JENKINS/Build+Failure+Analyzer
[6] https://wiki.jenkins-ci.org/display/JENKINS/External+Resource+Dispatcher
[7] https://wiki.jenkins-ci.org/display/JENKINS/Team+Views+Plugin

Table 2: Keywords used to classify commits

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		

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 can be seen in the Appendix.

Table 3: Interviewees demographics

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

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 he was contacted, had just left Sony Mobile for a smaller company 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 relevant findings during the interview wherever needed. The two first authors were present during all five interviews, with the addition of the third author during the first and fifth ones. Each interviewer took turns asking questions, whilst the others observed and took notes. Each interview was recorded and transcribed. A summary was also compiled and sent back to the interviewees for a review. Any misunderstandings or corrections could then be sorted out.

3.6 Validity Threats

This section highlights the validity threats related to the case study. Four types of validity threats [147] are addressed with their mitigation strategies.

Internal Validity

This concerns casual 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 firm's level to ensure that the innovation outcomes are merely the result of adopting OI. Therefore, more fined grained analysis on tools level helped us to minimize the threat of introducing confounding factors that might have caused innovation outcomes, which might not really be the consequence of adopting OI at Sony Mobile.

Subjectivity. It was found out 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 to infer the possible unconsciously used metrics. 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 commits messages classification criteria proposed by Hattori et al. [68]. 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 company

is a large scale organization with an intense focus on software development for embedded devices. Moreover, Sony Mobile is a direct competitor of all the main stream firms making Android phones. Like Sony Mobile, the involvement of 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.

Construct Validity

This deals with choosing the suitable measures for the concepts under study. We took the following measures 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 outcome by hiding facts or responding after assumed expectations. This threat was limited with the presence of a researcher that has a long research collaboration with Sony Mobile and explained confidentiality rules. Furthermore, interviewees were ensured complete anonymity both within the company and externally in the OSS community.

Design of the interviews. Multiple researchers validated the interview questionnaire followed by a pilot interview 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

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.

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 the classification of Sony Mobile's commit messages and commits per year (see Table 4). In Fig. 5, we depict the classification of commits for the seven repositories placed on the X-axis. The Y-axis depicts the classification of the content of the commits into the four categories while the bubble size corresponds to the number of commits. The Gerrit commits are divided into the core Gerrit (see Section 4.1) and the two plug-ins (Pygerrit, Gerrit-Event). The Jenkins commits are also divided into four plug-ins (Gerrit Trigger, Build failure analyzers, Ext. resource viewer and Team views). It should be pointed out that the commits related to Gerrit Trigger, Build failure analyzers, Ext. resource viewer and Team views belonged to Jenkins plug-ins and not the Jenkins core since it is comprised of 1000+ open plug-ins.

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 Fig.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 in March 2010 and as can be seen from the analysis also became active in contributions to Gerrit with 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.

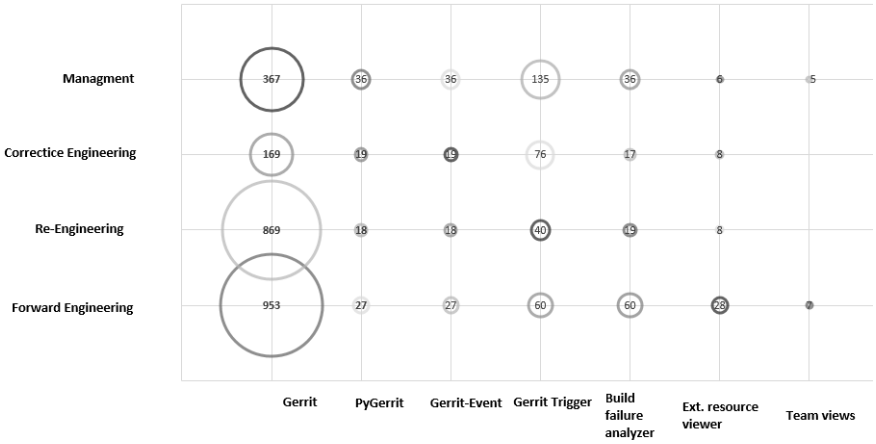


Figure 5: Bubble plot for Sony Mobile commits by classification

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 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 substantial number of forward engineering and re-engineering commits. In our opinion 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.

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

PyGerrit

PyGerrit is a Python library that provides a way for clients to interact with Gerrit Code Review via SSH or the REST API [8]. As can be seen in Table 5, 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

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

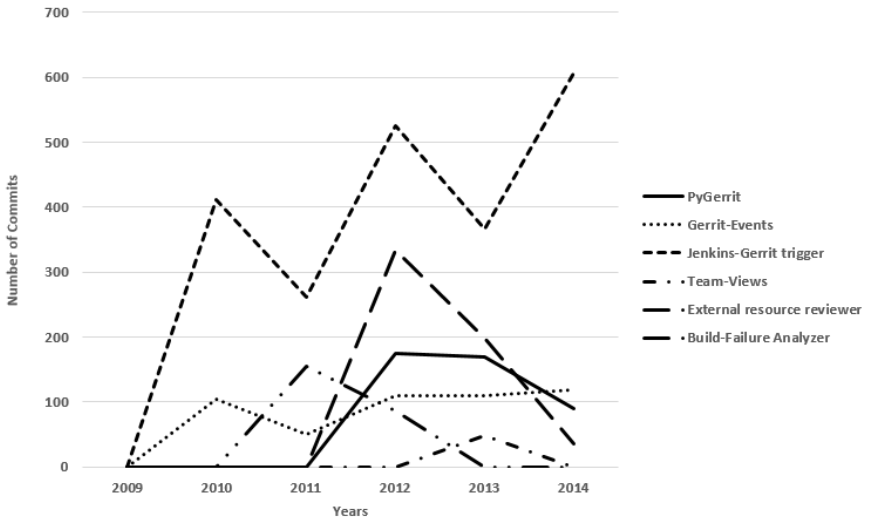


Figure 6: Sony Mobile's commits per year

Table 5: 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-dispatcher-pl	89.6	1.5	4.8	0	0	0	4.1
Build Failure Analyzer	85.5	0	0	0	0	0	14.4

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. This implies 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. Sony Mobile's yearly contribution analysis shows a steady growth since its introduction in 2011 (see Fig. 6).

Gerrit-Event

Gerrit-Event is a Java library used primarily to listen to stream-events from Gerrit Code Review and to send reviews via the SSH CLI or the REST API. It was originally a module in the Jenkins Gerrit Trigger plug-in and is now broken out to be used in other tools without the dependency to Jenkins. Table 5 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 5). Similarly, to the Pygerrit plug-in, management and forward engineering commits dominate and Sony Mobile is the main driver of features to this community.

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 5). The ones identified are:

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

Gerrit Trigger

This plug-in triggers builds on events from the Gerrit code review system by retrieving events from the Gerrit command "stream-events", so the trigger is pushed from Gerrit instead of pulled as scm-triggers usually are. Multiple builds can be triggered by one change-event, and one consolidated report is sent back to Gerrit. This plug-in (see Table 5) seem 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 was 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 a form of a new plug-in rather than extending the current plug-ins. This allows greater flexibility but increases the total number of parallel projects (plug-ins) 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 5, 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. See Section 5 for detailed analysis regarding this plug-in. Forward engineering and management commits are the two most common categories. Moreover, commits contribution have declined after 2012 and one possible explanation could be 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. Moreover, figure 5 shows the relatively low numbers of corrective engineering (17) and re-engineering (19) commits seem to indicate the maturity of this plug-in in terms of quality and functionality.

External Resource Viewer

This plug-in adds support for external resources in Jenkins. An external resource is something external attached to a Jenkins slave and can be locked by a build, which thus gets exclusive access to it, then released after the build is done. Examples of external resources are phones, printers and USB devices. 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. *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 views similar to a User's my-views. Sony Mobile turned out to be the only Committer for this tool (see Table 5), which implies that Team view is tailored for

Table 6: Themes emerging from the thematic analysis, and there mapping to RQs.

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

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 re-factoring was required for this plug-in. *This results also supports our previous hypothesis that modular plug-in based OSS communities are an efficient way for proprietary companies to participate and contribute with new functionality as new plug-ins.* 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 [32, 33] to find recurring patterns in the collected qualitative data. The following steps were performed in the process.

1. Transcribe the interviewed data
2. Identify and define five distinct themes (see Table 6) in the data
3. Classify the interviewed statements based on the defined themes
4. Summarize the findings and answers to the RQs

Furthermore, these themes were also mapped to the research questions in Table 6, which are further presented in the following subsections.

5.1 Opening up

This theme is related to *RQ2*, see Table 6. The process of opening up for external collaboration and maturing as an open source company, can be compared to

moving from a closed innovation model to an OI model [25]. 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 [172]. 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 IP rights, which could represent competitive advantages. The legal department took time in understanding the benefits and license aspects, which caused the initial contribution process to be extra troublesome. In this case, Sony Mobile benefited of having an internal champion and OSS evangelist (I5). 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 in an open forum. The OSS contribution process now includes submitting a form for a review, which promotes it further after successful initial screening. Next, the OSS review board gives it a go or no-go decision. As this would prove bureaucratic if it would be needed for each and every contribution to an OSS community, frame-agreements are created for open source projects with a high-intensity involvement, e.g. Jenkins and Gerrit. This creates a simplified and more sustainable process allowing for a day to day interaction between developers in the Tools department and the communities surrounding Jenkins and Gerrit.

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

This theme is related to *RQ3*, see Table 6. Several factors interplay in the decision process of whether or not a feature or a new project should be made open. Jenkins and Gerrit are neither seen as a part of Sony Mobile's competitive advantage nor as a revenue source. This is the main reason why developers in the Tools department can meet with competitors, go to conferences, give away free work etc. This, in turn, builds a general attitude that when something is about to be created, the question asked beforehand is if it can be made open source. There is also a follow-up question, whether Sony Mobile would benefit anything from it, for example maintenance, support and development from an active community. If a feature or a project is too specific and it is deemed that it will not gain any traction, the cost of generalizing the project for open use is not motivated. Another factor is whether there is an existing community for a feature or a project. By contributing a plug-in to the Jenkins community or a feature to Gerrit there is a chance that an active workforce is ready to adopt the contribution, whilst for new projects this has to be created from scratch which may be cumbersome.

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

This theme is related to *RQ5* (see Table 6) and provides insights about requirements engineering practices in an example OI context. The requirements process in the Tools department towards the Jenkins and Gerrit communities does not seem "very rigid", which is a common characteristic for OSS [149]. 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 their product development may post feature requests. However, a majority of the feature requests are submitted via e-mail. The developers in the Tools department started arranging monthly workshops where they invited the power users and the personnel from different functional roles in the product development organization. An open discussion is encouraged allowing for people to express their wishes and issues. An example of an idea sprung out from this forum is the Build Failure Analyzer[9] plug-in. Most of the requirements are, however, elicited internally within the Tools department in a dialogue between managers, architects and developers. They are seen to have the subject matter expertise in regards to the tool functionality. According to I2, there are *"... architect groups which investigate and collaborate with managers about how we could take the tool environment further"*. This is formulated as focus areas, and *"... typical examples of these requirements are sync times, push times, build times and apart from that everything needs to be faster and faster"*. These requirements are high level and later delegated to the development team for refinement.

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

The internal prioritization is managed by the development team itself, on delegation from the upper manager, and lead by two developers which have the assigned role of tool managers for Jenkins and Gerrit respectively. The focus areas frame the areas which need extra attention. Every new feature is prioritized against existing issues and feature requests in the backlog. External feature requests to OSS projects managed by the Tools department (e.g. the Gerrit Trigger 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.

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

An example regards the Gerrit Trigger plug-in and the implementation of different trigger styles. Pressing issues in the Tools department backlog kept them from working on the new features. At the same time, another software intense organization with interest in the plug-in contacted the Tools department about features they wanted to implement. These features and the trigger style functionality required larger architectural reconstruction. It was agreed that the external organization would perform the architectural changes with a continuous discussion with the Tools department. This allowed for a smaller workload and possibility to implement this feature earlier. This feature-by-feature collaboration is a commonly occurring practice as highlighted by I1, *"It's mostly feature per feature. It could be [Company] wants this feature and then they work on it and we work on it". But we don't have any long standing collaborations"*. I3 elaborates on this further and states that *"... its quite common for these types of collaboration to happen just between 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 [121]. As brought up by I2, *"... with the community you need to get people to look at it the same way as you do and get an agreement, otherwise it will be just discussions forever"*. This is extra problematic in the Gerrit community as the inner core team with the merge rights consists of only six people, of which one is from Sony Mobile. One of the key features received from the community was the tagging support for patch sets. I2 stated, *"... When developers upload a change which can have several revisions, it enabled us to tag meta-data like what is the issue in our issues handling system and change in priorities as a result of that change. This tagging feature allows the developers to handle their work flow in a better way"*. This whole feature was proposed and integrated during a hackathon, and contained more than 40 shared patch sets. Prior to implementing this feature together with the community (I3 quoted) *"... we tried to do it with the help of external consultants but we could not get it in, but meeting core developer in the community did the job for us"*.

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

This close collaboration with the community is important as Sony Mobile does

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

Conclusion: The requirements engineering process does not seem 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 Process

This theme highlights the testing process aspects associated with *RQ5*, see Table 6. 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, *"...Firms 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 [37]. 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 using Selenium, 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 department. It was decided to rework the framework to a Java-based version, which has helped the testing to take off although there still remains a lot to be done.

I3 highlighted that Sony Mobile played an important 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 new stable version. From the community perspective I3 stated, *"an Acceptance Test Harness also helps the community and other firms to understand what problems that big firms have or small firms in terms of features or in terms other requirements on the system. So it's a tool where everyone helps each other"*.

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

5.5 Innovation Outcomes

This theme is related to *RQ4* (see Table 6) where process and product innovation are highlighted [106]. In terms of measuring process and product innovation outcomes, Sony Mobile does not have any metrics. However, valuable insights were

found during the interviews regarding what Sony Mobile has gained from the Jenkins and Gerrit community involvement. During the analysis, following innovation outcomes are identified in the study.

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 [36, 156]. 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 there requires a higher effort in lobbying and actively steering the community in order to maximize the benefits for the organization. Along also comes, the ***free maintenance*** and quality assurance work, which renders better quality in the tools. Consequently, other than product innovations in tools as Jenkins and Gerrit, ***freed up time*** may be spent on other matters of high importance to Sony Mobile.

Correlated to the ***free work*** is the acknowledgement that the development team of six people in the Tools department will have a hard time keeping up with the external workforce, if they were to work in a closed environment. *"...I mean Gerrit has like letâĂŹs say we have 50 active developers, it's hard for the tech company to compete with that kind of workforce and these developers at Gerrit are really smart guys. It is hard to compete for commercial firms"*. 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 [121,156]. Otherwise, the knowledge tied to I3 would have risked being lost for Sony Mobile.

Before the paradigm shift and opening up of Sony Mobile, many tools were proprietary. Adapting these tools, such as the build server Electric commander, was cumbersome and it took long time before even a small fix would be implemented and delivered by the supplier. This created a stiffness whereas open source brought ***flexibility***. I2 quoted, *"...Say you just want a small fix, and you can fix that yourself very easily but putting a requirement on another company, I mean it can take years. Nothing says that they have to do it"*. This increase in the ***turnaround speed*** was besides the absence of license fees, a main argument in the discussions when looking at Jenkins as an alternative to Electric commander. 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 [121].

I1 stated that besides the flexibility, the Tools department is currently able to make a *"...more stable tools environment in [Sony Mobile] and that sort of makes our customers of the tools department, the testers and the engineers, to have an environment that actually works and does not collapse while trying to use it"*. I2 mentioned that *"...I think 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 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 company with multiple product branches. By opening up in the way that the Tools department has done, effects from the coupled OI processes with Jenkins and Gerrit may spread even

further into other product branches, possibly rendering in further innovations on different abstraction levels [106]. 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 [167]. The tool chain is even seen as the foundation for a platform which is supposed to facilitate this sharing [105]. 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"*.

In 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, inners source initiatives and faster time to market. However, these conclusions were drawn based on the qualitative data since Sony Mobile does not have any metrics to quantitatively measure the innovation outcomes.

6 Results and Discussion

Results from the quantitative and qualitative analysis are discussed below per theme.

6.1 Opening Up

The move to Android took Sony Mobile from a closed context to an external arena for OI reminds the description provided by Grotnes [64]. 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 [170] 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 broken down into a number of outside-in and inside-out transactions. The Tools department's involvement in Jenkins and Gerrit and the associated contribution process are repetitive and bidirectional. Thus, this interaction can be classified as a coupled innovation process [58]. This also complies with Grotnes' description of how an open membership renders in a coupled process, as Jenkins and Gerrit communities both are free for anyone to join, compared to the Android platform and its Open Handset Alliance, which is invite-only [64].

The quantitative results provide further support for the hypothesis that both incumbents and small scale software organizations are involved in the development of Jenkins and Gerrit (see Table 5). Some of the small organizations are

Garmin, Ostrovsky, Luksza, Codeaurora, Quelltextlich etc. This confirms findings from the existing OI literature, e.g. [71, 154] that other community players can also 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. Incumbents exploit communities to influence not only the development direction, but also to gain a good reputation in the community as underlined by prior studies [37, 71].

Gaining a good reputation requires more than just being an active Committer. Stam [154] 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 commits strategy. As described in Section 5.2, the intent was originally to keep the Gerrit Trigger plug-in enclosed. This form of selective revealing [70] 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. [175]. 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 [78]. 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 [71], and evangelize about open source was a great success factor, also identified in the inner source literature [111].

6.2 Determinants of Openness

When discussing if something should be made open or closed, 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 [102].

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 [163]. Wnuk et al. [175] reasons in a similar manner in their study where they differentiate between contributing early or late to the community in regards to specific features. By going with the former strategy, one may risk losing the competitive edge, however the latter creates potentially high maintenance costs.

Sony Mobile is aware of the fact that increased mobility [25] 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 [25]. 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 [103] Sony Mobile can use their contributions to steer and influence the direction of the community.

6.3 Requirements Engineering and Open Innovation

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) [149]. This applies both internally (as a supplier and an administrator of the tools), and externally (as a member of the communities). From the RE perspective, they are their own stakeholder, competing with other stakeholders (members) in Jenkins and Gerrit communities. These are important characteristics as stakeholders who are not developers themselves are often neither identified nor considered [3]. A consequence otherwise could be that certain areas are forgotten or neglected which stands in contrast to Wnuk et al. [175] 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. RE in OI applied through OSS hence can be seen as quicker, light-weight and more technically oriented than traditional RE [149].

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 [37]. Due to their high commitment and good track record, Sony Mobile employees have gotten high up 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 [154] (e.g. developer conferences [88]). One reason for this is to attract quiet stakeholders, both in terms of influencing the community [36], 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 [54] between the different stakeholders, as well as with grouping, can be deemed very ad-hoc as OSS RE is in general [149]. This correlates to the power structure and how influence may move between different stakeholders.

Social interaction between the stakeholders is stressed as an important aspect to resolve conflicts and to coordinate dependencies in distributed software development projects [137]. The Tools department preference for live meetings over the otherwise available electronic options such as mailing lists, issue trackers and discussion boards, is due to time differences and lag in discussions that complicate implementation of larger features. Open source hackathons [150] is the preferable choice as it brings the core stakeholders together which allows for informal negotiations [54] and a live just-in-time requirements process [45], meaning that requirements are captured in a less formal matter and first fully elaborated during implementation. As highlighted in Section 5.3, feature-by-feature collaborations is also a common practice. This is also due to the ease in communication as it may be performed between two single parties. Hence, it may be concluded that communication in this type of distributed development is a critical challenge, and in this case overcome by live meetings and keeping the number of collaborators per feature low.

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, firms 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 stake-

holders fluctuate depending on their respective interests. This objective and short-term way of looking at collaborations imply a need of standardized practices in a community for it to be effective. Furthermore, it highlights the need to continuously analyze the present stakeholders in the community in order to identify those with similar interests, both for possible collaborations and to find partners in order to gain leverage in prioritization processes, for example.

6.4 Testing process and Open Innovation

Both Jenkins and Gerrit focus on manual test cases. At the same time, the communities became 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. Seeing from this perspective, a trend may be visualized in how the different communities are becoming more professionalized in the sense that the tools make up business critical assets for many of the stakeholders in the communities, which motivates a continuous effort in risk-reduction [70, 124].

The move towards automated testing also allowed for the Tools department to contribute their internal test cases. This may be viewed as profitable from two angles. First, it reduces work load internally, and second, it secures that settings and cases specific for Sony Mobile are addressed and cared for. The test cases may to some extent be viewed as a set of informal requirements, which secure quality aspects in regards to scalability for example which is important for Sony Mobile [14]. Similar practices, but much more formal, are commonly used in more traditional (closed) software development environments. From a community perspective, other stakeholders benefit from this as they get the view and settings from a large environment which enable them to grow as well.

As can be noted in Fig. 5, the focus is on forward and re-engineering. An interested concern is when and how much one should contribute in regards to bug fixes and what should be left for the community, because some bug fixes are very specific to Sony Mobile and the community will not gain anything from 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.

6.5 Innovation Outcomes

The focal point of the OI theory is value creation and capture [24]. In the studied case, the value is created and captured through their involvement in the Jenkins and Gerrit communities. However, measuring that value using key performance indicators is a daunting challenge. Edison et al. [42] 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, interviewees mentioned that Sony Mobile does not have established mechanisms in place to measure their performance before and after the Jenkins and Gerrit introduction. This confirms the findings of Edison et al. [42]. However, from the qualitative data collected from the interviews we specifically looked for two types of innovations, product innovations in the tools Jenkins and Gerrit, and process innovation in Sony Mobile's product development. Other types, specifically market and organizational innovation were considered but not identified. Further, we did not differentiate between different impacts of innovation, such as incremental or radical [57].

By taking an active part in the knowledge sharing and exchange process with communities [36, 156], 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 [42]. This may also be viewed from the process innovation perspective [1] as Sony Mobile gets access to extra work force and a broad variety of competencies, which are internally unavailable [36]. 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 [94] *"No matter who you are, not all smart people work for you"*. Hence, it is vital to reach work force beyond organizations boundaries when innovating [24]. And as earlier described, knowledge is still retained as people move around inside the community.

Furthermore, these software improvements and product innovations affect the performance and quality of the continuous integration process used by Sony Mobile's product development. Continuous integration as an agile practice [9] enables early identification of integration issues as well as increases the developers' productivity and release frequency [153]. With this reasoning, as reported elsewhere [106], we deem that the product innovations captured in Jenkins and Gerrit transfer on as process innovation to Sony Mobile's product development. The main reason behind this connection is viewed as the possibility to tailoring and flexibility that the OSSD permits. By adapting the tool chain to the specific needs of the product development the mentioned benefits (e.g. increased build quality

and performance) are achieved and waste is reduced in the form of freed up hours, which product developers and testers may spend on alternative tasks as confirmed by Moller [120]. Reduced time to market and increased quality of products are among the visible business outcomes. However, these outcomes can not 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 [1] 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 [105]) 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. [121]. 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 [121]. Since Sony Mobile is a multinational corporation with a wide spread of internal culture, organizational changes are context and challenging.

6.6 Openness of Tools Software vs. Product software

An important finding of this study related to openness is that Sony Mobile only opens up its non-competitive tools that are not the part of the revenue stream. I3 stated that *"... Sony Mobile has learnt that even collaborating with its worst competitors does not take away their competitive advantage, rather they bring help for Sony Mobile and becomes better and better"*. Consequently, Sony Mobile's internal development environment has become more stable and gained access to the skilled workers from the community that would otherwise have come with significant costs. As a result, the communities receive commits from a large scale software organization. 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 performed on the area [127].

In the light of the mapping study conducted prior to this study, it would be fair to state that the SE literature lacks studies on OI [127]. 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 [162]. 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 [123]. 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 has focused on OI in SE at two levels: 1) innovation incorporated into Jenkins and Gerrit as software products, and 2) how these software improvements affect 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 firms' product or service offering, e.g. as a platform, component or full product [163]. 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 resources on the inside to gain influence, which together with an openness toward direct competitors and communities is used to draw attention to issues relevant for Sony Mobile, e.g. scalability of tools to large production environments. Social presence outside of online channels is highly valued in order to manage communication challenges related to distributed development. Another way of tackling such challenges regards co-creation on a feature-by-feature basis between two single parties. Choice of partner fluctuates and depends on 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 towards the pressing needs of the moment. Regarding testing, much focus is directed towards automating test activities in order to raise quality standards and professionalize communities to company standards.

Findings of the study are limited to software organizations with similar context, domain and size as Sony Mobile. It is also worth mentioning that stakeholdersâĂŹ involvement in Jenkins and Gerrit suggest that their continuous integration process is comparable to Sony Mobile thus, we believe that findings of this study may also be applicable to incumbents as well as small software 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 A

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

HOW FIRMS ADAPT AND INTERACT IN OPEN SOURCE ECOSYSTEMS: ANALYZING STAKEHOLDER INFLUENCE AND COLLABORATION PATTERNS

Johan Linåker, Patrick Rempel, Björn Regnell, Patrick Mäder

Abstract

[**Context and motivation**] Ecosystems developed as Open Source Software (OSS) are considered to be highly innovative and reactive to new market trends due to their openness and wide-ranging contributor base. Participation in OSS often implies opening up of the software development process and exposure towards new stakeholders. [**Question/Problem**] Firms considering to engage in such an environment should carefully consider potential opportunities and challenges upfront. The openness may lead to higher innovation potential but also to frictional losses for engaged firms. Further, as an ecosystem progresses, power structures and influence on feature selection may fluctuate accordingly. [**Principal ideas/results**] We analyze the Apache Hadoop ecosystem in a quantitative longitudinal case study to investigate changing stakeholder influence and collaboration patterns. Further, we investigate how its innovation and time-to-market evolve at the same time. [**Contribution**] Findings show collaborations between and influence shifting among rivaling and non-competing firms. Network analysis proves valuable on how an awareness of past, present and emerging stakeholders, in regards to

power structure and collaborations may be created. Furthermore, the ecosystem's innovation and time-to-market show strong variations among the release history. Indications were also found that these characteristics are influenced by the way how stakeholders collaborate with each other.

1 Introduction

The paradigm of Open Innovation (OI) encourages firms to look outside for ideas and resources that may further advance their internal innovation capital [25]. Conversely, a firm may also find more profitable incentives to open up an intellectual property right (IPR) rather than keeping it closed. For software-intensive firms a common example of such a context is constituted by Open Source Software (OSS) ecosystems [82, 170].

The openness implied by OI and an OSS ecosystem makes a firm's formerly closed borders permeable for interaction and influence from new stakeholders, many of which may be unknown to a newly opened-up firm. Entering such an ecosystem affects the way how Requirements Engineering (RE) processes are structured [107]. Traditionally these are centralized, and limited to a defined set of stakeholders. However, in this new open context, RE has moved to become more decentralized and collaborative with an evolving set of stakeholders. This may lead to an increased innovation potential for a firm's technology and product offerings, but also imply frictional losses [36]. Conflicting interests and strategies may arise, which may diminish a firms own impact in regards to feature selection and control of product planning [175]. Further, as an ecosystem evolves, power structures and influence among stakeholders may fluctuate accordingly. This creates a need for firms already engaged or thinking of entering an OSS ecosystem to have an awareness of past and present ecosystem governance constellation in order to be able to adapt their strategies and product planning to upcoming directions of the ecosystem [80].

Given this problematization, we were interested in studying how stakeholders' influence and collaboration fluctuate over time in OSS ecosystems. Researchers argue that collaboration is core to increase innovation and reduce time-to-market [44]. Hence, another goal was to study the evolution of OSS ecosystems' innovation and time-to-market over time. We hypothesize that this could be used as input to firms' planning of contribution and product strategies, which led us to formulate the following research questions:

RQ1 How are stakeholder influence and collaboration evolving over time?

RQ2 How are innovation and time-to-market evolving over the same time?

To address these questions, we launched an exploratory and quantitative longitudinal case study of the Apache Hadoop ecosystem, a widely adopted OSS framework for distribution and process parallelization of large data.

The rest of the paper is structured as follows: Section 2 presents related work. Section 3 describes the case study design and methodology used, limitations and threats to validity are also accounted for. Section 4 presents the analysis and results, which are further discussed in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Here we present related work to software ecosystems and how its actors (stakeholders) may be analyzed. Further, the fields of stakeholder identification and analysis in RE are presented from an ecosystem and social network perspective.

2.1 Software Ecosystems

Multiple definitions of a software ecosystem exists [114], while we refer to the one by Jansen et al. [82] - *"A software ecosystem is a set of actors functioning as a unit and interacting with a shared market for software and services, together with relationships among them. These relationships are frequently underpinned by a common technological platform or market and operates through the exchange of information, resources and artifacts."*. The definition may incorporate numerous types of ecosystems in regards to openness [81], ranging from proprietary to OSS ecosystems [114], which in turn contains multiple facets. In this study we will focus on the latter with the Apache Hadoop ecosystem as our case, where the Apache Hadoop project constitutes the technological platform underpinning the relationships between the actors of the Apache Hadoop ecosystem.

An ecosystem may further be seen from three scope levels, as proposed by Jansen et al. [80]. Scope level 1 takes an upper perspective, on the relationships and interactions between ecosystems, for example between the Apache Hadoop and the Apache Spark ecosystems, where the latter's project may be built on top of the former. On scope level 2, one looks inside of the ecosystem, its actors and the relationships between them, which is the focus of this paper when analyzing the Apache Hadoop ecosystem. Lastly, scope level 3 takes the perspective from a single actor and its specific relationships.

Jansen et al. [80] further distinguished between three types of actors: dominators, keystone players, and niche players. Dominators expand and assimilate, often on the expense of other actors. Keystone players are well connected, often with a central role in hubs of actors. They create and contribute value, often beneficial to its surrounding actors. Platform suppliers are typically keystone players. Niche players thrive on the keystone players and strive to distinguish themselves from other niche players. Although other classifications exist [114] [81], we will stick to those defined above.

In the context of OSS ecosystems, a further type of distinction can be made in regards to the Onion model as proposed by Nakakoji et al. [128]. They dis-

tinguished between eight roles ranging the passive user in the outer layer, to the project leader located in the center of the model. For each layer towards the center, influence in the ecosystem increases. Advancement is correlated to increase of contributions and engagement of the user, relating to the concept of meritocracy.

2.2 Stakeholder Networks and Interaction in Requirements Engineering

To know the requirements and constraints of a software, one needs to know who the stakeholders are, hence highlighting the importance of stakeholder identification and analysis in RE [61]. Knowing which stakeholders are present is however not limited to purposes of requirements elicitation. For firms engaged in OSS ecosystems [82, 114], this is important input to their product planning and contribution strategies. Disclosure of differentiating features to competitors, un-synced release cycles, extra patch-work and missed out collaboration opportunities are some possible consequences if the identification and analysis of the ecosystem's stakeholders is not done properly [36, 170, 175]. Most identification methods however refer to the context of traditional software development and lack empirical validation in the context of OSS ecosystems [136].

In recent years, the research focus within the field has shifted more towards stakeholder characterization through the use of, e.g., Social Network Analysis (SNA) [136]. It has also become a popular tool in empirical studies of OSS ecosystems, hence highlighting potential application within stakeholder identification.

In regards to traditional software development, Damian et al. [38] used SNA to investigate collaboration patterns and the awareness between stakeholders of co-developed requirements in the context of global software development. Lim et al. [104] constructed a system based on referrals, where identified stakeholders may recommend others. Concerning RE processes within software ecosystems in general, research is rather limited [54] with some exceptions [88]. Fricker [54] proposed that stakeholder relations in software ecosystems may be modeled as requirement value chains *" ... where requirements emerge from and propagate with inter-stakeholder collaboration"*. Knauss et al. [88] investigated the IBM CLM ecosystem to find RE challenges and practices used in open-commercial software ecosystems. Distinction is made between a strategic and an emergent requirements flow, where the former regard high level requirements, and how business goals affect the release planning. The latter considers requirements created on an operational level, in a Just-In-Time (JIT) fashion, commonly observed in OSS ecosystems [45].

In OSS ecosystems specifically, RE practices such as elicitation, prioritization, and selection are usually managed through open forums such as issue trackers or mailinglists. These are also referred to as informalisms as they are used to specify and manage the requirements in an informal manner [149], usually as a part of a conversation between stakeholders. These informalisms constitute an

important source to identify relevant stakeholders. Earlier work includes Duc et al. [41] who applied SNA to map stakeholders in groups of reporters, assignees, and commentators to issues with the goal to investigate the impact of stakeholder collaboration on the resolution time of OSS issues. Crowsten et al. [31] performed SNA on 120 OSS projects to investigate communication patterns in regards to interactions in projects' issue trackers.

Many studies focused on a developer and user level, though some exceptions exist. For example, Martinez-Romeo et al. [116] investigated how a community and a firm collaborates through the development process. Orucevic-Alagic et al. [135] investigated the influence of stakeholders on each other in the Android project. Texiera et al. [158] explored collaboration between firms in the Openstack ecosystem from a co-opetition perspective showing how firms, despite being competitors, may still collaborate within an ecosystem.

This paper contributes to OSS RE literature by addressing the area of stakeholder identification and analysis in OSS ecosystems by investigating a case on a functional level [158]. Further it adds to the software ecosystem literature and its shallow research of RE [54, 88] and strategic perspectives [114] in general.

3 Research Design

We chose the Apache Hadoop project for an embedded case study [146] due to its systematically organized contribution process and its ecosystem composition. Most of the contributors have a corporate affiliation.

To create a longitudinal perspective, issues of the Apache Hadoop's issue tracking and project management tool were analyzed in sets reflecting the release cycles. The analysis was narrowed down to sub releases, spanning from 2.2.0 (released 15/Oct/13) to 2.7.1 (06/Jul/15), thus constituting the units of analysis through the study. Third level releases were aggregated into their parent upper level release.

Issues were furthermore chosen as the main data source as these can tie stakeholders' socio-technical interaction together [38, 41], as well as being connected to a specific release. To determine who collaborated with whom through an issue, patches submitted by each stakeholder were analyzed, a methodology similar to those used in previous studies [116, 135]. Users who contribute to an issue package their code into a patch and then attach it to the issue in question. After passing a two-step approval process comprising automated tests and manual code reviews, an authorized committer eventually commits the patch to the project's source configuration management (SCM) system. The overall process of this case study is illustrated in Fig. 1 and further elaborated on below.

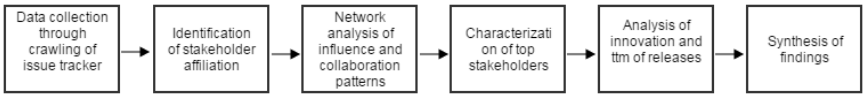


Figure 1: Overview of the case study process

3.1 Data Collection

The Apache Hadoop project manages its issue data with the issue tracker JIRA. A crawler was implemented to automatically collect, parse, and index the data into a relational database.

To determine the issue contributors' organizational affiliation, the domain of their email addresses was analyzed. If the affiliation could not be determined directly (e.g., for @apache.org), secondary sources were used such as LinkedIn and Google. The issue contributors' full name functioned as keyword.

3.2 Analysis Approach and Metrics

Below we present the methodology and metrics used in the analysis of this paper. Further discussion of metrics in relation to threats to validity is available in section 6.

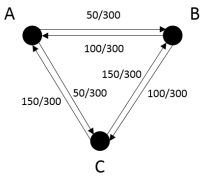


Figure 2: Example of a weighted network with three stakeholders.

Network Analysis. Patches attached to issues were used as input to the SNA process. Stakeholders were paired if they submitted a patch to the same issue. Based on stakeholders' affiliation, pairings were aggregated to the organizational level. A directed network was constructed, representing the stakeholders at the organizational level as vertices. Stakeholder collaboration relationships were represented as edges. As suggested by Orucevic-Alagic et al. [135], edge weights were calculated to describe the strength of the relationships. Since stakeholders created patches of different size, the relative size of a stakeholder's patch was used for the weighting. We quantified this size as changed lines of code (LOC) per patch. A simplified example of calculating network weights without organizational aggregation is shown in Fig. 1. Each of the stakeholders A, B, and C created a patch that was attached to the same issue. A's patch contains 50 LOC. B's patch contains 100 LOC, while C's patch contains 150 LOC. In total, 300 LOC were contributed to the

issue. Resulting in the following edge weights: A→B = 50/300, A→C = 50/300, B→C = 100/300, B→A = 100/300, C→B = 150/300, and C→A = 150/300.

The following network metrics were used to measure the influence of stakeholders and the strength of the collaboration relationships among the stakeholders.

- *Out-degree Centrality* is the sum of a all outgoing edges' weights of a stakeholder vertex. Since it calculates the number of collaborations where the stakeholder has contributed, a higher index indicates a higher influence of a stakeholder on its collaborators. It also quantifies the degree of contributions relative to the stakeholder's collaborators.

- *Betweeness Centrality* counts how often a stakeholder is on a stakeholder collaboration path. A higher index indicates that the stakeholder has a more central position compared to other stakeholders among these collaboration paths.

- *Closeness Centrality* measures the average relative distance to all other stakeholders in the network based on the shortest paths. A higher index indicates that a stakeholder is well connected and has better possibilities in spreading information in the network, hence a higher influence.

- *Average Clustering Coefficient* quantifies the degree to which stakeholders tend to form clusters (connected groups). A higher coefficient indicates a higher clustering, e.g., a more densely connected group of stakeholders with a higher degree of collaborations.

- *Graph Density* is the actual number of stakeholder relationships divided by the possible number of stakeholder relationships. A higher value indicates a better completeness of stakeholder relationships (collaborations) within the network, where 1 is complete and 0 means that no relationships exist.

Innovation and Time-To-Market Analysis. Innovation can be measured through input, output, or process measures [89]. In this study, input and output measures are used to quantify innovation per release. Time-to-market was measured through the release cycle time [63].

- *Issues* counts the total number of implemented JIRA tickets per release and comprises the JIRA issue types *feature*, *improvement*, and *bug*. It quantifies the innovation input to the development process.

- *Change size* counts the net value of changed lines of code. It quantifies the innovation output of the development process.

- *Release cycle time* is the amount of time between the start of a release and the end of a release. It indicates the length of a release cycle.

Stakeholder Characterization. To complement our quantitative analysis and add further context, we did an qualitative analysis of electronic data available to characterize identified corporate stakeholders. This analysis primarily included their respective websites, press releases, news articles, and blog posts.

3.3 Threats to Validity

Four aspects of validity in regards to a case study are *construct*, *internal* and *external validity*, and *reliability,runeson2009guidelines*.

In regards to *construct validity*, one concern may be definition and interpretation of network metrics. The use of weights to better represent a stakeholder's influence, as suggested by Orucevic-Alagic et al. [135] was used with the adoption to consider the net of added LOC to further consider the relative size of contributions. A higher number of LOC however does not have to imply increased complexity. We chose to see it as a simplified metric of investment with each LOC representing a cost from stakeholder. Other options could include consideration software metrics such as cyclomatic complexity. Further network metrics, e.g. the eigenvector centrality and the clustering coefficient could offer further facets but was excluded as a design choice.

Furthermore, we focused on input (number of issues) and output (implementation change size) related metrics [89] for operationalizing the innovation per release. Issues is one of many concepts in how requirements may be framed and communicated in OSS RE, hence the term requirement is not always used explicitly [149]. Types of issues varies between OSS ecosystem and type of issue tracker (e.g., JIRA, BugZilla) [45]. In the Apache Hadoop ecosystem we have chosen the types feature, improvement and bug to represent the degree of innovation. We hypothesize that stakeholders engaged in bug fixing, are also involved in the innovation process, even if a new feature and an improvement probably includes a higher degree of novelty in the innovation. Even bugs may actually include requirements-related information not found elsewhere, and also relate to previously defined features with missing information. In future work, weights could be introduced to consider different degrees of innovation in the different issue types.

Release cycle times were used for quantifying the time-to-market as suggested by Griffin [63]. Since we solely analyzed releases from the time where the Apache Hadoop ecosystem was already well established, a drawback is that a long requirements analysis ramp up time may not be covered by this measure.

A threat to *internal validity* concerns the observed correlation of how the time-to-market and the innovativeness of a release is influenced by the way how stakeholders collaborate with each other. This needs further replication and validation in future work.

In regards to *external validity*, this is an exploratory single case study. Hence observations need validation and verification in upcoming studies in order for findings to be further generalized. Another limitation concerns that only patches of

issues were analyzed, though it has been considered a valid approach in earlier studies [116, 135]. In future work, consideration should also be taken into account, for example, as this may also be an indicator of influence and collaboration. Further, number of releases in this study was limited due to a complicated release history in the Apache Hadoop project, but also a design choice to give a further qualitative view of each release in a relative fine-grained time-perspective. Future studies should strive to analyze longer periods of time.

Finally, in regards to *reliability* one concern may be the identification of stakeholder affiliation. A contributor could have used the same e-mail but from different roles, e.g., as an individual or for the firm. Further, sources such as LinkedIn may be out of date.

4 Analysis

In this section, we present our results of the quantitative analysis of the Apache Hadoop ecosystem across the six releases R2.2-R2.7.

4.1 Stakeholders' Characteristics

Prior to quantitatively analyzing the stakeholder network, we qualitatively analyzed stakeholders' characteristics to gain a better understanding of our studied case. First, we analyzed how each stakeholder uses the Apache Hadoop platform to support its own business model. We identified the following five user categories:

- **Infrastructure provider**: sells infrastructure that is based on Apache Hadoop.
- **Platform user**: uses Apache Hadoop to store and process data.
- **Product provider**: sells packaged Apache Hadoop solutions.
- **Product supporter**: Provides Apache Hadoop support without being a product provider.
- **Service provider**: Sells Apache Hadoop related services.

Second, we analyzed stakeholders' firm history and strategic business goals to gain a better understanding of their motivation for engaging in the Hadoop ecosystem. We summarize the results of this analysis in the following list:

- **Wandisco** [Infrastructure provider] entered the Apache Hadoop ecosystem by acquiring AltoStar in 2012. It develops a platform to distribute data over multiple Apache Hadoop clusters.
- **Baidu** [Platform user] is a web service company and was founded in 2000. It uses Apache Hadoop for data storage and processing of data.

- **eBay** [Platform user] is an E-commerce firm and was founded in 1995. It uses Hadoop for data storage and processing of data.

- **Twitter** [Platform user] offers online social networking services and was founded in 2006. It uses Apache Hadoop for data storage and processing of data.

- **Xiaomi** [Platform user] is focused on smartphone development. It uses Apache Hadoop for data storage and processing of data.

- **Yahoo** [Platform user] is a search engine provider who initiated the Apache Hadoop project in 2005. It uses Apache Hadoop for data storage and processing of data. It spun off Hortonworks in 2011.

- **Cloudera** [Product provider] was founded in 2008. It develops its own Apache Hadoop based product *Cloudera Distribution Including Apache Hadoop* (CDH).

- **Hortonworks** [Product provider] was spun off by Yahoo in 2011. It develops its own Apache Hadoop based product *Hortonworks Data Platform* (HDP). It collaborates with Microsoft since 2011 to develop *HDP for Windows*. Other partnerships include Redhat, SAP, and Terradata.

- **Huawei** [Product provider] offers the Enterprise platform *FusionInsight* based on Apache Hadoop. FusionInsight was first released in 2013.

- **Intel** [Product supporter] maintained its own Apache Hadoop distribution that was optimized to their own hardware. It dropped the development in 2014 to support Cloudera by becoming its biggest shareholder and focusing on contributing its features to Cloudera's distribution.

- **Altiscale** [Service provider] was founded in 2012. It runs its own infrastructure and offers Apache Hadoop as-a-service via their product *Altiscale Data Cloud*.

- **Microsoft** [Service provider] offers Apache Hadoop as a cloud service labeled *HDInsight* through its cloud platform Azure. It maintains a partnership with Hortonworks who develops *HDP for Windows*.

- **NTT Data** [Service provider] is a partner with Cloudera and provides support and consulting services for their Apache Hadoop distribution.

Firms that belong to the same user category apply similar business models. Hence, we can identify competing firms based on their categorization.

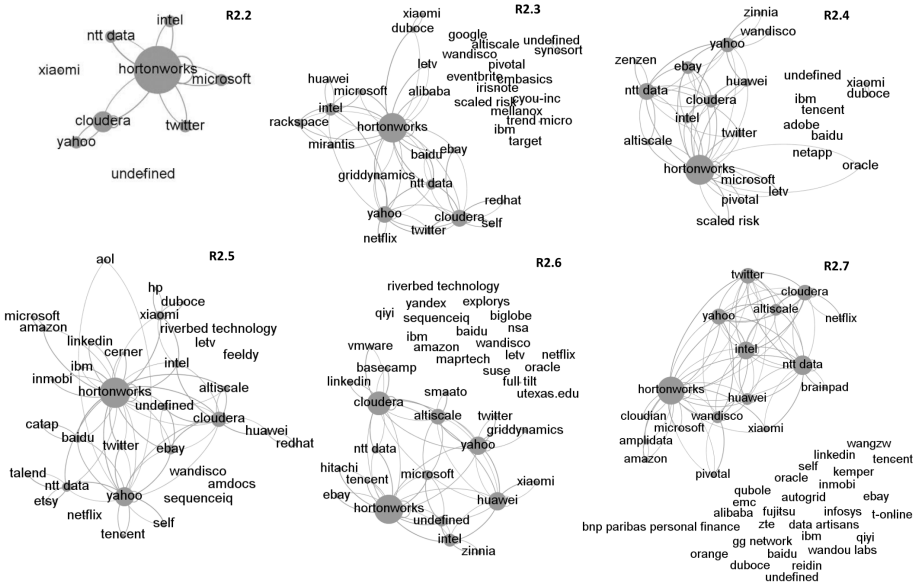


Figure 3: Network distribution of releases R2.2-R2.7

Table 1: Number of stakeholder (vertices) and collaboration relationships (edges) per release

	R2.2	R2.3	R2.4	R2.5	R2.6	R2.7
Stakeholders	9	35	25	34	38	44
Collaboration relationships	21	97	81	108	96	122

4.2 Stakeholder Collaboration

Figure 3 shows all stakeholder networks that were generated for the releases R2.2 to R2.7. The size of a stakeholder vertex indicates its relative ranking in regards to the outdegree centrality. Table 1 summarizes the number of stakeholders and stakeholder relationships per release. It illustrates that the number of stakeholders and collaboration relationships varies over time. Except for the major increase from R2.2 to R2.3, the network maintains a relatively consistent size, though the number of collaborations are in the interval between 81 to 122 for R2.4 to R2.7.

A general observation among the different releases is the existence of one main cluster where a core of stakeholders is present, whilst the remaining stakeholders make temporary appearances. Many stakeholders are not part of these clusters implying that they do not collaborate with other stakeholders at all. The number of those stakeholders shows strong variation among the releases. This could imply that stakeholders implement their own issues, which is further supported by the

fact that 65% of the patches are contributed by the issue reporters themselves.

The visual observation from the networks being weakly connected in general is supported by the Graph Density (GD) as its values are relatively low among all releases (see Table 2). The values describe that stakeholders had a low number of collaborations in relation to the possible number of collaborations. The Average Clustering Coefficient (ACC) values among all releases (see Table 2) further indicate that the stakeholders are weakly connected to their direct neighbors in the releases R2.2 - R2.6. This correlates with the observation that there are many unconnected stakeholders and only a few core stakeholders collaborating with each other. The ACC value however indicates a significantly higher number of collaborations for release R2.7.

Table 3 summarizes stakeholder collaborations among the different user categories. It shows that collaborations took place among all user categories, except between infrastructure providers and service providers. The product providers were the most active and had the highest number of collaborations with other product providers. They also have the highest amount of collaborations with other user categories. These results show that stakeholders with competing (same user category) and non-competing (different user category) business models collaborate within the Apache Hadoop ecosystem.

4.3 Stakeholder Influence

To analyze the evolving stakeholder influence over time, we leveraged the three network centrality metrics: outdegree centrality, betweeness centrality, and closeness centrality.

The left graph in Fig. 4 shows the outdegree centrality evolution for the ten stakeholders with the highest outdegree centrality values. These stakeholders are most influential among all Apache Hadoop stakeholders in regards to weighted issue contributions. The graph also shows that the relative outdegree centrality varies over time. To further investigate this evolution, we created a stakeholder ranking per release using the relative outdegree centrality as ranking criteria. This analysis revealed that Hortonworks was most influential in terms of issue contributions. It was five times ranked first and once ranked third (average ranking: 1.3). The other top ranked stakeholders were Cloudera (average ranking: 3.3) and Yahoo (average ranking: 3.3). The stakeholders NTT Data (avg ranking = 4.7) and Intel (average ranking: 4.8) can be considered as intermediate influencing among the top ten outdegree centrality stakeholders. The stakeholders Huawei (average

Table 2: Average Clustering Coefficient (ACC) and Graph Density (GD) per release.

	R2.2	R2.3	R2.4	R2.5	R2.6	R2.7
ACC	0	0.207	0.303	0.198	0.237	0.552
GD	0.292	0.082	0.135	0.096	0.068	0.064

Table 3: Stakeholder collaborations among the different user categories.

	Infrastructure provider	Platform user	Product provider	Product supporter	Service provider
Infrastructure provider	0	2	4	1	0
Platform user	2	24	73	6	14
Product provider	4	73	124	23	50
Product supporter	1	6	23	0	3
Service provider	0	14	50	3	10

ranking: 8.2), Twitter (average ranking: 8.5), eBay (average ranking: 9.0), Microsoft (average ranking: 9.5), and Baidu (average ranking: 10.2) had the least relative outdegree centrality among the ten stakeholders.

The center graph in Fig. 4 shows the betweeness centrality evolution of the ten stakeholders with the highest accumulated values. As the metric is based on the number of shortest paths passing through a stakeholder vertex, it indicates a stakeholder's centrality with regards to the possible number of collaborations. The resulting top ten stakeholder list is very similar to the list of stakeholders with the highest outdegree centrality. The top stakeholders are Hortonworks (average ranking: 1), Cloudera (average ranking: 2.7), and Yahoo (average ranking: 3.0). Intel (average ranking: 4.2), NTT Data (average ranking: 4.7), and Huawei (average ranking: 5.3) are influencing among the top ten beweeness centrality stakeholders. eBay (average ranking: 6.7), Amazon (average ranking: 6.7), WANdisco (average ranking: 7.0), and Baidu (average ranking: 7.2), the group of stakeholders with the least betweeness centrality among the top ten stakeholders differs compared to the

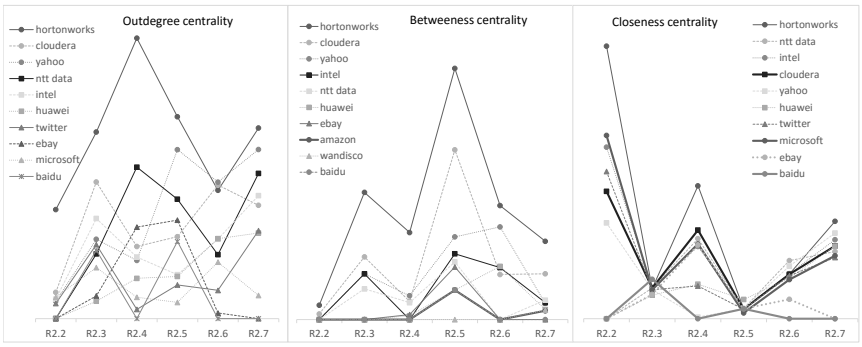


Figure 4: Evolution of stakeholders' outdegree, betweeness, and closeness centrality across the releases R2.2-R2.7

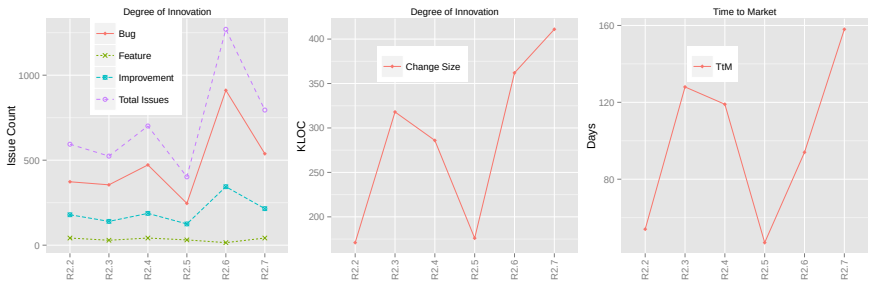


Figure 5: Evolution of the degree of innovation over time with respect to implemented JIRA issues and changed lines of code and time to market.

group of stakeholders with the least outdegree centrality. The stakeholders Twitter and Microsoft were replaced by Amazon and WANdisco.

The right graph in Fig. 4 shows closeness centrality evolution of the ten stakeholders with the highest accumulated values. A higher degree of closeness centrality indicates higher influence, because of closer collaboration relationships to other stakeholders. The resulting top ten closeness centrality stakeholder list differs compared to the outdegree and betweeness centrality list. Our analysis results do not show a single top stakeholder with the highest closeness centrality. The stakeholders Hortonworks (avgerage ranking: 3.2), NTT Data (average ranking: 4.0), Intel (average ranking: 4.3), Cloudera (average ranking: 4.8), and Yahoo (average ranking: 5.5) had relatively similar closeness rankings among the releases. This is also reflected in Fig. 4 by very similar curve shapes among the stakeholders. Also the remaining stakeholders with lower closeness centrality values had very similar average rankings: Huawei (average ranking: 7.7), Twitter (average ranking: 8.0), Microsoft (average ranking: 8.3), eBay (average ranking: 9.2), Baidu (average ranking: 9.3).

The results of our analysis also show that the stakeholders with the highest outdegree centrality, betweeness centrality, and closeness centrality were distributed among different stakeholder user categories: 4 platform user, 3 product provider, 2 service provider, and 1 product supporter. However, it is notable that the average ranking differs among these user categories. Product providers had the highest average influence ranking. Platform users and service providers had lower influence ranking. This implies that product providers are the most driving forces of the Apache Hadoop ecosystem.

4.4 Innovation and Time-To-Market over Time

The evolution of the degree of innovation and time-to-market from release R2.2 to R2.7 is summarized in Figure 5 by three consecutive graphs. The first graph in Fig. 5 shows the number of issues that were implemented per release. The illus-

trated number of issues is broken down into the issue types: bug, improvement, and feature. The number of implemented features (avg: 33.5, med: 37, std: 9.88) remains steady across all analyzed releases. This is reflected by a relatively low standard deviation. Similarly, the number of implemented improvements (avg: 198.3; med: 183; std: 71.62) remains relatively steady across the releases with one exception. In release R2.6, the double amount of improvement issues was implemented compared to the average of the remaining releases. The number of implemented bugs (avg: 482.5; med: 423; std: 212.52) features stronger variation among the releases.

The second graph in Fig. 5 shows the number of changed lines of code per release. The total number of changed lines of code per release (avg: 287,883.33; med: 302,257; std: 89,334.57) strongly varies across the analyzed releases. Each of the analyzed releases comprises code changes of significant complexity. Even the two releases R2.2 and R2.5, with the lowest change complexity (R2.2: 171 KLOC; R2.5: 176 KLOC), comprised more than 170 KLOC. The remaining releases comprised change complexities of more than 250 KLOC. Further, the graph indicates that the change complexity scatters randomly among the studied releases. A steady trend cannot be determined.

The third graph in Fig. 5 depicts the time between the start and the end (time-to-market) of each analyzed release. Analogous to the evolution of the changed lines of code, the time-to-market scatters randomly among the analyzed releases.

5 Discussion

Stakeholder Collaborations (RQ-1). The number of collaborating stakeholders remains on a relatively stable level. However, as indicated by the GD and ACC, the networks are weakly connected in regards to the possible number of collaborations. Only a core set of stakeholders is engaged in most of the collaborations. This may indicate that they have a higher stake in the ecosystem with regards to their product offering and business model, and in turn a keystone behaviour [80]. From a requirements value chain perspective, collaborations translate into partnerships and relationships. This may prove valuable in negotiations about requirements prioritization and how these should be treated when planning releases and road maps [54]. The results also show that many stakeholders do not collaborate at all. This is supported by the fact that 65% of the reported issues are implemented by reporters themselves without any collaboration. This indicates that a lot of independent work was performed in the ecosystem. Reasons for this could be that issues are only of interest for the reporter. It also indicates that the ecosystem is relatively open [81] in the sense that it is easy for stakeholders to get their own elicited requirements implemented and prioritized, but with the cost of own development efforts.

Another aspect of the collaborations can be inferred from the different user categories. Firms with competing business models collaborate as openly as non-rivaling firms do, as presented in Table 3 and reported in earlier studies [158]. Some of the collaborations may be characterized through the partnerships established between the different stakeholders, as presented in our qualitative analysis of stakeholder characteristics. One of Hortonworks many partnerships include that with Microsoft through the development of their Windows-friendly Apache Hadoop distribution. Cloudera's partnerships include both Intel and NTT Data. None of these partnerships, or among the others identified in this study, occurs within the same user category. Yet still, a substantial part of the ecosystem collaboration occurs outside these special business relationships.

Independent of business model, all firms work together towards the common goal of advancing the shared platform, much resembling an external joint R&D pool [170]. As defined through the concept of co-opetition, one motivation could be a joint effort to increase the market share by helping out to create value, and then later diverge and capture value when differentiating in the competition about the customers [129]. Collaboration could further be limited to commodity parts whereas differentiating parts are kept internal, e.g. leveraged through selective revealing [72].

Stakeholder Influence (RQ-1). Although the distribution of stakeholders' influence fluctuated among the releases, we identified that the group of most influential stakeholders remained very stable. Even the influence ranking within this group did not show high variations. It can be concluded that the development is mainly driven by the stakeholders Hortonworks, Cloudera, NTT Data, Yahoo, and Intel, which may also be referred to as keystone players, and in some cases also niche players relative to each other [80]. Due to this stable evolution, it can be expected that these stakeholders will also be very influential firms in the future. The stakeholder distribution represents multiple user categories, although the product providers Hortonworks and Cloudera tend to be in the top. This may relate to their products being tightly knit with the Apache Hadoop project. In turn, service-providers may use the product-providers' distributions as a basis for their offerings.

Tracking that influence may be useful to identify groups and peers with key positions in order to create traction on certain focus areas for the road map, or to prioritize certain requirements for implementation and release planning [54]. Further, it may help to identify emerging stakeholders increasing their contributions and level of engagement [128], which may also be reflected in the commercial market. Huawei's increase in outdegree centrality, for example, correlates with the release of their product FusionInsight, which was launched in the beginning of 2013.

The fact that the network metrics used revealed different top stakeholders, indicates the need of multiple views when analysing the influence. For example, the

betweeness centrality Xiaomi, Baidu, and Microsoft in the top compared to the outdegree centrality. This observation indicates that they were involved in more collaboration but produced lower weighted (LOC) contributions relative to their collaborators.

Evolution of Ecosystem in Regards to Innovation and Time-To-Market (RQ-2). The analysis results indicate that the number of implemented features does not vary among the analyzed releases. A possible reason for this could be the ecosystem's history. From release R2.2 to R2.5, the project was dominated by one central stakeholder (Hortonworks). Although, additional stakeholders with more influence emerged in release R2.6 and R2.7, Hortonworks remained the dominating contributor, who presumable continued definition and implementation of feature issues. Another potential reason for the lack of variance among features could be the fact that our analysis aggregated all data of third level minor releases to the upper second level releases.

However, our results indicate that the number of implemented improvements show variations among the releases. From release R2.2 to R2.5, the number of implemented improvements per release remained at a steady level. For release R2.6 and R2.7, the number of implemented improvements increased (double the amount). A possible reason for the observed effect could be the fact that other stakeholders with business models get involved in the project to improve the existing ecosystem with respect to their own strategic goals that helps to optimally exploit for their own purpose. The number of implemented bugs varies among all analyzed releases. The high variance of the number of defects could be a side effect of the increased number of improvement issues that potentially imply increase in overall complexity within the ecosystem. Further, the more stakeholders get actively involved in the project to optimize their own business model the more often the ecosystem is potentially used, which may increase the probability to reveal previously undetected defects.

The analysis results with respect to the evolution of the change size indicate a strong variance among all analyzed releases. Similarly to the change size, the time-to-market measure showed great variance among the analyzed releases. Covariances of stakeholder collaboration, degree of innovation, and time-to-market measure among the analyzed releases may indicate relationship between these variables. However, to draw this conclusion a detailed regression analysis of multiple ecosystems is required.

Implications for Practitioners. Even though an ecosystem may have a high population, its governance and project management may still be centered around a small group of stakeholders [128], which may further be classified as keystone and in some cases, niche players. Understanding their evolving composition and the influence of these stakeholders may indicate current and possible future directions of the ecosystem [80]. Corporate stakeholders could use this information to better

align their open source engagement strategies to their own business goals [158]. It could further provide insights for firms, to what stakeholders' strategic partnerships should be established to improve their strategic influence on the ecosystem regarding, e.g., requirement elicitation, prioritization and release planning [54]. Here it is of importance to know how the requirements are communicated throughout the ecosystem, both on a strategic and operational level for a stakeholder to be able to perform the RE processes along with maximized use of its influence [88]. Potential collaborators may, for example, be characterized with regards to their commitment, area of interest, resource investment and impact [62].

The same reasoning also applies for analysis of competitors. Due to the increased openness and decreased distance to competitors implied by joining an ecosystem [80], it becomes more important and interesting to track what the competitors do [36]. Knowing about their existing collaborations, contributions, and interests in specific features offer valuable information about the competitors' strategies and tactics [158]. The methodology used in this study offers an option to such an analysis but needs further research.

Knowledge about stakeholder influence and collaboration patterns may provide important input to stakeholders' strategies. For example, stakeholders may develop strategies on if or when to join an OSS ecosystem, if and how they should adapt their RE processes internally, and how to act together with other stakeholders in an ecosystem using existing practices in OSS RE (e.g., [45, 149]). This regards both on the strategic and operational level, as requirements may be communicated differently depending on abstraction level, e.g., a focus area for a road map or a feature implementation for an upcoming release [88]. However, for the operational context in regards to how and when to contribute, further types of performance indicators may be needed. Understanding release cycles and included issues may give an indication of how time-to-market correlates to the complexity and innovativeness of a release. This in turn may help to synchronize a firm's release planning with the ecosystem's, minimizing extra patchwork and missed feature introductions [175]. Furthermore, it may help a firm planning their own ecosystem contributions and maximize chances for inclusion. In our analysis, we found indications that the time-to-market and the innovativeness of a release is influenced by the way how stakeholders collaborate with each other. Hence, the results could potentially be used as time-to-market and innovativeness predictors for future releases. This however also needs further attention and replication in future research.

6 Conclusions

The Apache Hadoop ecosystem is generally weakly connected in regards to collaborations. The network of stakeholders per release consists of a core that is continuously present. A large but fluctuating number of stakeholders work independently. This is emphasized by the fact that a majority of the issues are implemented by the issue reporters themselves. The analysis further shows that the network maintains

an even size. One can see that the stakeholders' influence as well as collaborations fluctuate between and among the stakeholders, both competing and non-rivaling. This creates further input and questions to how direct and indirect competitors reason and practically work together, and what strategies are used when sharing knowledge and functionality with each other and the ecosystem.

In the analysis of stakeholders' influence, a previously proposed methodology was used and advanced to also consider relative size of contributions, and also interactions on an issue level. Further, the methodology demonstrates how an awareness of past, present and emerging stakeholders, in regards to power structure and collaborations may be created. Such an awareness may offer a valuable input to a firm's stakeholder management, and help them to adapt and maintain a sustainable position in an open source ecosystem's governance. Consequently, it may be seen as a pivotal part and enabler for a firm's software development and requirements engineering process, especially considering elicitation, prioritization and release planning for example.

Lastly, we found that innovation and time-to-market of the Apache Hadoop ecosystem strongly varies among the different releases. Indications were also found that these factors are influenced by the way how stakeholders collaborate with each other.

Future research will focus on what implications stakeholders' influence and collaboration patterns have in an ecosystem. How does it affect time-to-market and innovativeness of a release? How does it affect a stakeholder's impact on feature-selection? How should a firm engaged in an ecosystem adapt and interact in order to maximize its internal innovation process and technology advancement?

CHAPTER IV

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

Johan Linåker, Hussan Munir, Krzysztof Wnuk, Carl-Eric Mols

Abstract

Open Source Software (OSS) ecosystems have reshaped 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 base on OSS. Existing models accurately capture the 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 aligns 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 classifying 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 within the firm's organization. The CAP model provides the necessary structure and operational support for OSS ecosystem participation and facilitates strategic product planning in regards to what and when to contribute.

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 [169] but OI remains unexplored in software engineering [127]. A notable exception however is that of Open Source Software (OSS) ecosystems [80,168,170]. By adopting OSS as part of a firm's business model [26] (directly or indirectly), OSS may help the firm to accelerate its internal innovation process [24]. 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 [156, 163]. A further potential benefit is the inflow of innovative 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"*.

However, to better realize these potential benefits of OI resulting from participation in OSS ecosystems, firms need to establish synchronization mechanisms between their product strategy realization and business models and their ecosystem participation and roles [127,154,175]. The central part of this synchronization is to know *what to differentiate* and *what commodity parts* to use in the product. A wrong focus may imply unnecessary internal maintenance and patch-work, un-synced release cycles and the give-away of valuable functionality to competitors [175]. A common practice is to contribute parts considered as commodity, while focusing on keeping differentiating parts closed [70,168]. The timing aspect is critical here as functionality sooner or later will pass over from being differentiating to commodity due to a constantly progressing technology life cycle [162].

Achieving the above mentioned synchronization between the product strategy and product planning [87] allows for strategic product planning in OI and helps to realize the OI potential. Further, this could help firms to answer questions such as what parts that can be considered contributable, and when. As proposed by previous work [175], we choose to define such guidelines that explain *what* can be contributed, and *when* as a contribution strategy. The existing commoditization models [17, 162] are not designed with the specifics of OSS ecosystem participation in mind and therefore lack support for strategic product planning and contribution strategies. Despite the potential positive consequences of contribution strategies for firms, software engineering literature lacks evidence for firms that have established such strategies and alignment between their product strategies and related OSS ecosystems [127].

This paper occupies this research gap by presenting a Contribution Acceptance Process (CAP) model developed in close collaboration with Sony Mobile that is actively involved in a number of OSS ecosystem, both in regards to their products

features and internal development infrastructure such as development tools. Based on an extensive investigation of Sony Mobile's contribution processes and policies, we propose the Contribution Acceptance Process (CAP) model designed to support strategic product planning by aligning product strategies and contribution strategies towards OSS ecosystems. The CAP model is an important step for firms that use OSS ecosystems in their product development and want to gain or increase the potential benefits of OI. Moreover, we propose the information meta-model that helps to instantiate the CAP model in a firm that stores information about product strategy, requirements, architecture assets, patches and contributions.

The rest of the paper is dispositioned 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 how these were managed. 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. Lastly, in section 6 we discuss the CAP model in relation to related work, and specific considerations, while we summarize our study in section 7.

2 Related Work

Below we describe the context of our research in regards to how software engineering and OSS fits into the OI context. Further, we describe what a contribution strategy is, how it can be explained in relation to earlier research, and its connection to software artifacts. Lastly, we summarize by describing the research gap, which this study aims to fill.

2.1 Open Innovation in Software Engineering

OI is commonly explained by a model based on a funnel [25] which in our case represents the firm and its internal software development process, see Fig. 1. The funnel is permeable, meaning that the firm can interact with the open environment surrounding it, in our case, an OSS ecosystem. These interactions are represented by the arrows going in and out, and can be further characterized as transactions and exchange of knowledge between the firm and the OSS ecosystem. Examples of transactions can include software artifacts (e.g., big fixes, feature implementations, plug-ins, or complete projects), but also opinions, knowledge and support that could regard any step of the internal or external development.

The illustrated interactions 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*). When outside-in and inside-out transactions occurs together, the processes are termed *coupled innovation* [44]. This may be expected in co-development between a firm and other ecosystem participants in regards to specific functionality.

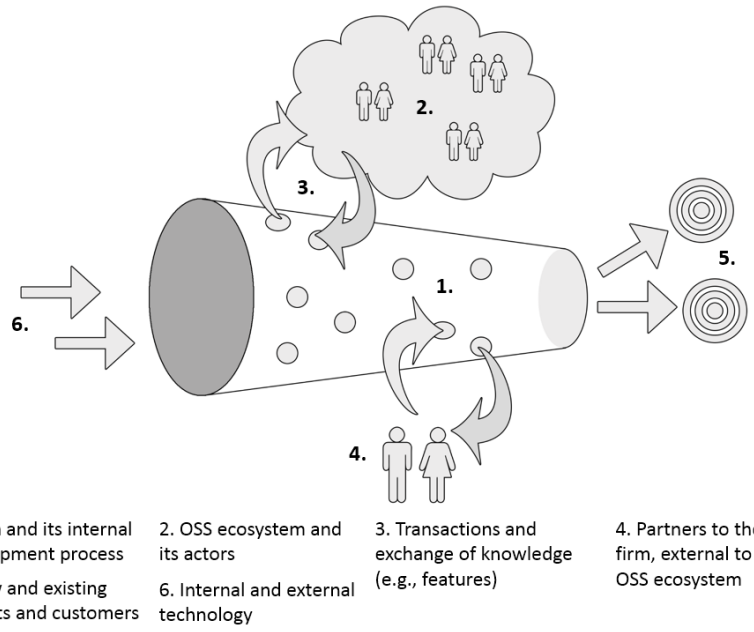


Figure 1: The OI model illustrated with interactions between the firm (funnel) and its external collaborations. Adopted from Chesbrough [25].

2.2 Contribution Strategies in Open Source Software

Wnuk et al. [175] define a contribution strategy as a managerial practice that helps to decide what to contribute as OSS, and when. This is a fundamental tool in the requirements scoping process of a firm, further defined by Wnuk et al. [175] as *"... a process of deciding: (1) which version of the OSS product should be utilized, (2) what should be added to the platform in order to create product differentiation and competitive advantage, (3) what shall be contributed to the OSS [ecosystem] (and when), and (4) how to influence the OSS [ecosystem] and become the leading stakeholder to maximize return on investment and reduce uncertainty"*. A well-established contribution strategy helps to bridge product planning and release planning with the commoditization process the forces the developed functionality to be returned to OSS ecosystems.

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 [36] describe how a firm may access the OSS ecosystems in order to extend the firms resource base, align the firm strategy with that of the OSS ecosystem and/or assimilate the OSS ecosystem in order to integrate and share results with them. In another study, Dahlander & Magnusson [35] continue and describe how a firm can adapt their relationship towards the OSS ecosystem based on how much influence they need, e.g., by openly contributing back to the OSS ecosystem, or keeping changes and new features internal. Based on how open a firm uses the OSS and its ecosystem in their business model and level of influence needed, different strategies may be applied. For example, selectively revealing means that differentiating parts are kept internal while commodity parts are contributed [70, 168]. Further, licenses may be used so that the technology can be disclosed under conditions where control is still maintained [168]. In the edge case, everything could be disclosed under open and transparent conditions [26], or even kept closed. As highlighted by Jansen et al. [81], openness of a firm should be considered as a continuum rather than a binary choice between open and closed.

It is important to recognize that contribution strategies affects and is affected by several functions in a firm. As with road mapping [92], all relevant stakeholders need to be involved to add their view on the software artifacts. This can further be exemplified as different roles in the firm are connected to different abstractions of software artifacts as they get broken down. For example, it may be decided that a set of features should be made OSS based on an evaluation of marketing. However, it may be that certain requirements needs a different classification due to architectural reasons as deemed by engineering. The CAP model presented in this study is intended to handle software artifacts of these different abstraction levels. Artifacts within or between an abstraction level can be further characterized, e.g., in regards to size or type as suggested by Hattori et al. [68]. The CAP model takes a different approach and proposes three different levels (minor, medium and major

contributions) based on Sony Mobile's internal practice and experience.

2.3 Commoditization Models

With commoditization models, we refer to a model that describes a software artifact's value and how it moves between a differential to a commodity state, i.e., if the artifact is considered to help distinguish the focal firm's product offering relative its competitors. Such models can help firms better understand what they should contribute, and when, i.e., provide a base to design contribution strategies from [175]. Van der Linden et al. [162] describe efficient software development as focusing *"...on producing only the differentiating parts"*. They continue that *"...preferably, firms acquire the commodity software elsewhere, through 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 lifecycle and more often experienced towards the end of it [87].

Van der Linden et al. [162] present a commoditization model that describes how software and technology moves from being differentiating, to being basic for business, and finally considered as commodity. In relation it should be considered whether the software or technology should be developed, acquired, or kept internally, between other firms, or made completely open (e.g., as OSS). Ideally differentiating software or technology are kept internally, but as their life-cycle progresses and move towards being a commodity, it is made more open. Bosch [17] presents a similar commoditization model, which also classifies the software into three layers but on a wider level. He 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, while finally transitioning to stage where it is considered as commodity, hence it *"...no longer adds any real value"* [17].

Bosch et al. [17] further emphasize how software from the three distinct layers of functionality change at different rates, and their intertwining leads to system complexity. Managing this complexity requires separation of these layers. As newer and older functionality become mixed, software architecture erodes over time. Change of a component in the system leads to overall structural changes in the multiple components. Therefore, Bosch suggest that these layers are separated architecturally. Decoupling differentiation functionality from the commodity functionality helps firms to easily replace the functionality with commercial or OSS solution. Further, by decoupling differential and experimental functionality, products can be tested and refined in the innovation and experimentation layer for market distribution.

A challenge identified by both underlying studies [17, 162] is the risk of losing Intellectual property rights (IPR) to competitors, which has also been highlighted

in other research [70, 71, 170, 175]. By not contributing software and technology that are considered differentiating, firms can avoid giving away its added value to competitors. In addition, acquiring the commodity functionality helps 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 [162].

2.4 Sourcing and Purchasing

Peter Kraljic was the first to outline a portfolio model of the stages of purchase sophistication and advocated that purchasing must become supply chain process with a clear strategy [93]. An important part of the model suggested by Kraljic is the strategic positioning phase where opportunity and vulnerability areas are identified and matched with supply risks and trust in suppliers. The Kraljic's portfolio model shows the firm's strength in purchasing assets against the supply market specifics. The model inspired several industries and academics. Among some examples, Caniëls and Gelderman [22] studied the choice of various purchasing strategies and empirically quantified the "relative power" and "total interdependence" aspects among Dutch purchasing professionals. caniels2005purchasing et al. looked at purchasing as a market shaping mechanism and identified five types of market shaping actions [160]. Shaya discussed the usage of the Kraljic's portfolio model for optimizing the process of sourcing IT and managing software licenses at Skanska ITN [152]. Gangadharan et al. proposed using Kraljic's portfolio model for mapping SaaS services and sourcing structure [56]. Their contribution includes a classification for sourcing SaaS products supported by sourcing structured on strategic, tactical and operational levels. However, their work is on the service level which is often the entire product level and does not consider OSS contributions a possible strategy. Our focus is on a feature level as a part of the product. To the best of our knowledge, no study has suggested using the Kraljic's portfolio model for supporting strategic product management in OI.

2.5 Strategic Product Planning in OI

Creating software product strategies and planning software products based on these strategies are the two main areas of a Software Product Manager's (SPM) responsibility [87]. The software product strategy should contain the product definition in terms of functional and quality scope, target market, delivery model, positioning and sourcing [1] The decision if a software product should be based on OSS from an OSS ecosystem should be made by executive management after the recommendation from the Software Product Manager [113]. The decisions regarding the scope of the project in relation to possible OSS realization strategies should be made by the SPMs, who are currently deprived of models that can

[1] http://community.ispma.org/wp-content/uploads/2015/05/ISPMA-SPM-FL-Syllabus-V-1.2.pdf

translate the selected product strategy and scope into operational decisions what to develop based on OSS and when to contribute the solution, or certain parts of it, back to the OSS ecosystems. Thus, we present a model for strategic product planning as it looks beyond realizing a set of features in a series of software releases that reflect the overall product strategy and adds the OI strategy aspect executed by OSS contribution content and timing.

2.6 Artifacts in Software Engineering

In the context of this study, an artifact refers to a requirement(s), a piece of code, frameworks as enablers for apps, test case(s) or related documentation. A common example would be a representation of the wish from a customer, which may be abstracted, restructured and specified in a series of different ways and levels, e.g., from being a business objective on a goal level, to being represented by a series of task descriptions, architectural-notes, design-suggestions, test cases and documentation. Artifacts may be put together as models and used as reference models to help capture specific results and to secure all necessary requirements, test cases, documentation linked together. Some of the common examples are Requirements Engineering Reference (REM) Model [59] and the IEEE software requirements specification Std. 830-1998 [29]. REM proposes three major types of artifacts: Business Needs Specification (e.g., business objectives, main features), the Requirements Specification (e.g., application scenarios, domain model) and the System Specification (e.g., data model, system interaction) [48].

Dependent on the context and process used, these artifacts may be structured and stored in different ways. Artifacts are often stored in a central repository and expected to meet certain quality criteria (e.g., in regards to completeness and traceability) [3]. In contrast, OSS ecosystems constitute an opposite extreme with their usually very informal practices [45]. Here, requirements may be specified in several ways, often complementing each other to give a fuller picture, e.g., as an issue on the projects issue tracker, in conversations on the projects mailing lists, and/or as a prototype or a finished implementation. However, prioritization of requirements and test cases are very much dependent on the individual needs of the firm [126, 127]. These are examples of what Scacchi refers to as informalisms [149].

3 Research methodology

In this section, we describe the research design and process of our study, as well as 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 threats to validity and how these were managed.

3.1 Case Firm

Sony Mobile is a multinational firm with roughly 5,000 employees, developing embedded devices. 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 [138].

3.2 Research Questions

Two research questions are investigated in this study. The first research question (**RQ1**) is focused on aligning contribution decisions with strategic product planning in OI. The trade-off that many software-intensive firms need to consider is how to maintain active OSS ecosystem involvement by frequent and significant contributions, and at the same time sustain competitive advantage and realize strategic product plans. In particular, we are focusing on providing tangible guidelines that helps product planners in understanding how long a feature or a set of features remains competitive advantage and when it should be contributed back to the OSS ecosystem. There is literature explaining general incentives and strategies for how firms should act [36, 72, 170], but the existing models [17, 162] does not consider aspects specific to OSS and firms' strategic product planning [127]. Hence, we pose our first research question as:

RQ1 How can contribution strategies be created and structured to support strategic product planning in OI?

The second research question focuses on developing a meta-model of information (repositories) that supports the alignment described in **RQ1** and helps decision makers in daily operations.

RQ2 What data sources are required and how they should be represented in a meta-model to enable strategic product planning and contribution strategy alignment?

3.3 Research Design and Operation

This study is a case study [147] with an influence of design science [73]. First there was a problem investigation to identify the research problem and confirm its relevance. Second, this was followed by an artifact design process where the artifacts addressing the problem was created. Finally, the artifacts were validated in how they addressed the research problem. These steps were performed in iteratively in close collaboration with the case firm. Throughout the steps, as illustrated in Fig. 2, data collection and analysis was performed, ending with the reporting of the results, which is constituted by this study.

Problem Identification

The problem identification process regarding both **RQ1** and **RQ2** was initiated by informal consultations with four experts of Sony Mobile who are involved in the decision making process of OSS contributions, see table1. Simultaneously, internal processes and policy documentation at Sony Mobile were studied. From the consultations, we received further support to access additional data sources and were able to investigate requirements databases, contribution databases and process documentation. These findings confirmed that a suitable solution has to be a combination of technology-based artifact (a contribution acceptance process model) and organization-based artifacts (an information structure) (see guidelines one and two by Hevner [73]).

Table 1: Consultation with experts

Expert Id	Years of experience	Role
I1	6 Years	Team Lead
I2	11 years	Director OSS SW Operations
I3	15 Years	Senior Manager
I4	5 Years	Software Developer

Artifacts Design

In order to address **RQ1**, we formalized our findings from the consultations with I1-4 and studies of internal processes and policy documentation, in a Contribution Acceptance Process (CAP) model supported by guidelines for using it for strategic product planning in OI. This was an iterative process where further consultations, especially with I2, was performed along with further document studies.

Further, to complement the CAP model and address **RQ2**, an information meta-model was created and derived from the software artifact repositories connected to the Android Platform used in Sony Mobile's products. These repositories covered their internal product portfolio, feature repository, feature-based architectural asset repository, patch repository, contribution repository and commit repositories (see Fig. 2). For simplicity, we narrowed our search process to the data for one of Sony Mobile's Android platforms and extracted all attributes from its repositories. All the relevant repositories for the selected platform with its complete trace can be seen in section 5.1. The identification process of repositories and collection of its data was done through consultation with I1-4. We ensured that the right balance between research rigor and relevance is kept and therefore moved away from extensive mathematical formalizations of the CAP model and focused on the applicability and generalizability of the model (see guideline five by Hevner [73]).

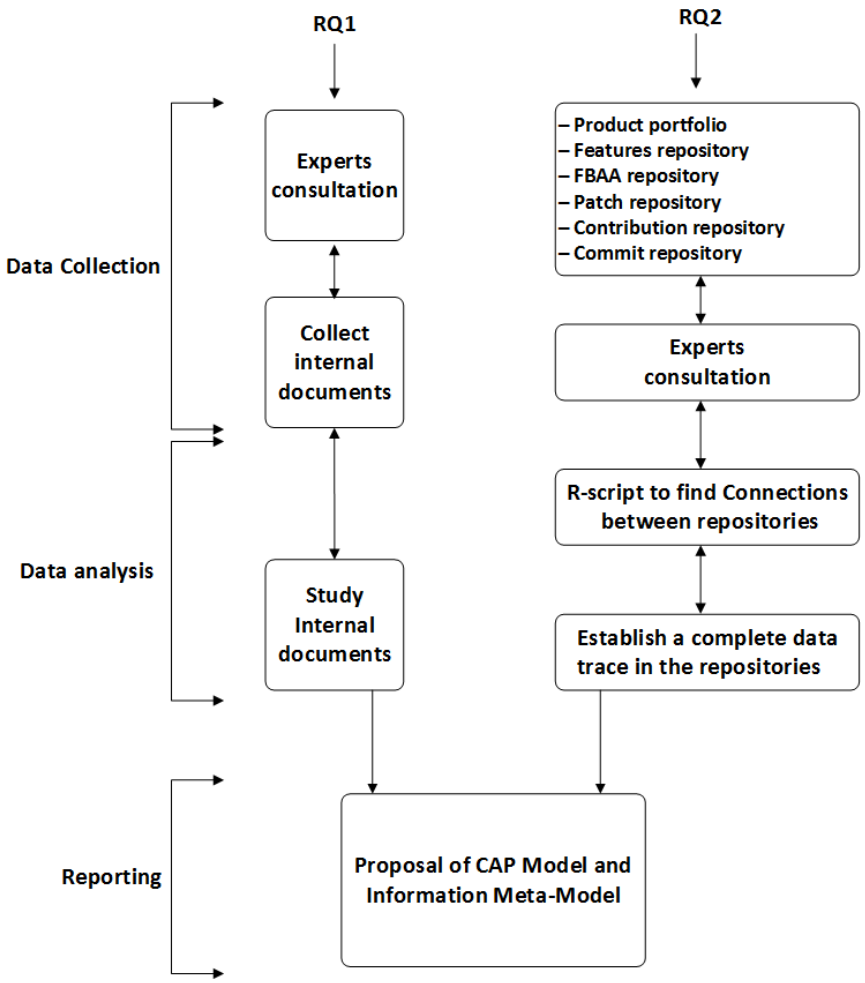


Figure 2: Overview of the research methodology used in this study. These methods were performed iteratively through the three steps involved in design science: problem investigation, artifact design, and artifact valuation [73].

The extracted data was additionally validated with the concerned experts (I1-4) at Sony Mobile through informal communication for the validation. We wrote R-scripts[2] in order to find traces between platforms, patches, contributions to OSS ecosystems and commits. We selected only those features which have a complete trace depicted in Fig. 4.

Artifacts Validation

The CAP-model and related information meta-model were validated statically through continuous consultations with experts at Sony Mobile (I1-4). Further, a series of real-world examples of software artifacts was used to describe how the CAP-model can be used to motivate what should be contributed and when. These examples were elicited applied together with I2 to evaluate functionality, completeness and consistency of the CAP model and associated information meta-model. The presented meta-model also concerns the implementability evaluation aspect and it confirms that the CAP model can be instantiated in an industrial context. Summarizing, we combined observational (case study) and descriptive evaluation where we obtained detailed scenarios to demonstrate the utility of the CAP model (see guideline three by Hevner [73]).

3.4 Ethics and Confidentiality

This study required analysis of sensitive data from Sony Mobile related to its Android phone development. The researchers in the study had the right to maintain their integrity and adhere to agreed procedures. Researchers arranged meeting with experts from Sony Mobile to inform them about the reporting of the study. Furthermore, data acquired from Sony Mobile were kept confidential and will not be shared publicly 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 the Sony Mobile (I2) 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 [147] are mentioned along with their mitigation strategies.

Internal Validity

Internal validity refers to factors affecting the outcome of the study without being in the knowledge of researchers [147].

[2]https://www.r-project.org/

Researchers bias. The proposed CAP model is 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 in the study. At the same time, the central part of the CAP model involves estimating complexity and business impact. 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 not in the direct focus of the CAP model.

Triangulation. Most of the analysis was driven by quantitative data obtained from Sony Mobile. In order to mitigate the risk of identifying the right data flows, concerned experts were consulted at Sony Mobile.

External Validity

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

We have focused on analytical generalization rather than statistical generalization [51] by comparing the characteristics of the case to a possible target and presenting case firm characteristics as much as confidentiality concerns allow to facilitate direct comparison to other cases. The scope of this study is limited to firms realizing OI with OSS ecosystems. The selected case firm represents organizations 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 on all firms involved in OSS ecosystems. It should be noted that the case firm can be considered as a mature firm in the regards that they see how they can make use of OSS to create product value and realize product strategies. Also, they recognize the need to invest resources in the ecosystems by contributing back in order to be able the influence and control in accordance to internal needs and incentives. Thus, the application of the proposed CAP model in other context or other firms remain the part of future work.

Construct Validity

Construct validity deals with choosing the suitable measures for the concepts under study [147]. In this study, there was a risk that academic researchers and industry practitioners may use different terms and have different theoretical frames of reference to address contribution strategies to OSS ecosystem. Furthermore, the presence of researcher may have threatened the experts from Sony Mobile to give information according to researchers' assumed expectations. The selection of a handful of experts from Sony Mobile might also contribute to the unbalanced

view of the construct. In this study, we used following mitigation strategies to counter the above mentioned risks.

Common theoretical frame of reference. In this study Kraljic's portfolio model is used as a framework of reference to propose CAP model. However, the horizontal and vertical dimensions of Kraljic's portfolio model are changed to control complexity and business impact respectively. Both industry practitioners and academic researchers had a common understanding of Kraljic's portfolio model [93] 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 same understanding.

Prolonged involvement. 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 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. To mitigate this risk, the first two researchers analyzed the data independently and discussed the proposed data flow and model with the remaining authors. In addition, multiple contacts from Sony Mobile were contacted to ensure the correctness of the data.

Audit trail. 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 Kraljic [93]. Kraljic's model was originally constructed to help firms with decision-support to how they should procure or source their production material. The CAP model is focused on software artifacts and how these can be sourced and shared as OSS. The artifacts may be of different abstraction levels, e.g., ranging from specific requirements or issues to set of requirements as features, frameworks, tools or complete products. For simplicity, we limit our focus to features in the presentation of the CAP model this study.

4.1 Model Structure

In the original version of the Kraljics's model, products and materials were classified in two dimensions: *profit impact* and *supply risk* (low and high). In the CAP model, these are replaced with the two metrics *business impact* and *control complexity*, see Fig. 3. Business impact refers to how much you profit from the artifact and is represented by the vertical axis. Control complexity refers to how hard the technology and knowledge behind the artifact is to acquire and control and is represented on the horizontal axis. Both metrics range from low to high.

To determine the business impact of the artifacts set of questions is used. The answers to these questions are given on a Likert scale with values between 1 and 5. The questions are as follows:

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?

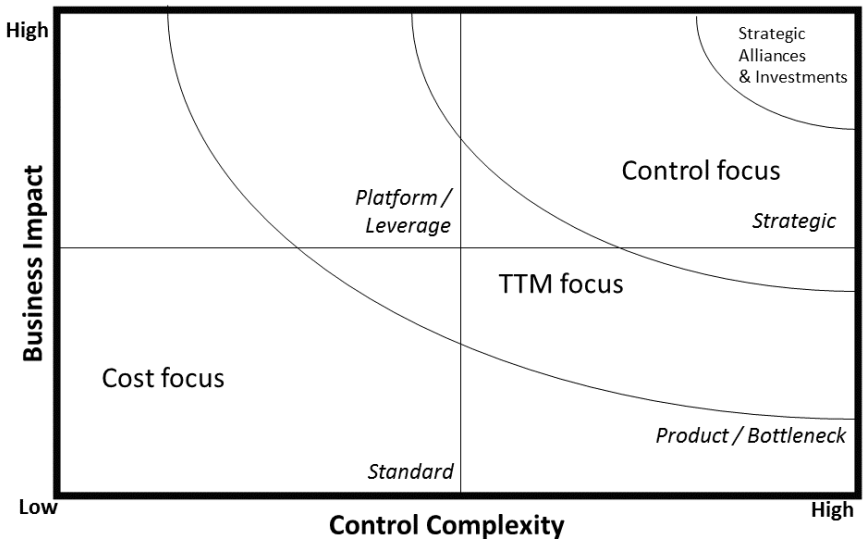
As with the business impact, a set of questions are asked in order to determine the control complexity of the artifacts on a scale between 1-5:

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?

Based on how the artifacts are mapped with respect to business impact and control complexity, the firm should take a specific focus which further helps them to choose a correct contribution strategy for the artifacts. Below we describe these focus areas as presented in Fig. 3.

- **Cost focus.** Features that are classified in the cost focus area provide little or no competitive advantage, i.e., they are considered as commodity or enablers for other features. In this case, the 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. By contributing as much as possible, internal maintenance may be reduced and resources may be shifted towards tasks has more differentiation value for a firm.

Figure 3: 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*.



- **Time-to-market (TTM) focus.** Features considered in this area contain substantial competitive advantage elements and bring substantial value to end customers. Thus, a firm that realizes these features need to decide when to release them and how long they should be differentiated from the common OSS project. The number of patches can initially be high but should be gradually reduced as the features get commoditized and competitors catch up.

- **Control focus.** Features that require the firm to gain and maintain a certain position in the OSS ecosystem's governance structure [5] in order to better manage conflicting agendas and to avoid costly adaptions to others' solutions.

- **Strategic Alliances and Investments.** Features in the top right corner of the CAP model carry a large part of product innovation and competitive advantage. Thus, they should be internally developed or co-developed using strategic alliances and investments that secure IPR ownership.

In the CAP model, software artifacts of different (or within the same) abstraction level(s) (e.g., requirements, features, framework), can be further characterized in one of three different levels:

Trivial contributions are rather small changes to already existing open source software, 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 regards 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 [125].

4.2 The Contribution Acceptance Process

The classification of software artifacts in regards to the CAP model should be performed by those who have the relevant knowledge. As in the roadmapping process [92], this is ideally done cross-functionally as marketing may best judge the priorities of the customers, while engineering may best judge how important an artifact is in regards to the software architecture. Other internal stakeholders may also have an opinion, e.g., legal department may have opinions on licenses. Depending on the size of the firm, these classification groups may be placed on different levels but should still maintain the cross-functional constellation in order to get the views from each of the internal stakeholders.

In Sony Mobile the proposed classification process is not yet operationalized. They do however have internal open source governance board, but foremost in the purpose of overseeing the contribution process and managing IPRs and possible copyright infringements. The board has a cross-functional composition as previously suggested with engineers, business managers and legal experts.

Engineers who want to make a contribution need to send a request through their business manager who makes a decision based on business benefits. For the trivial contributions, the business manager's approval is enough. For medium and major contributions however, the business manager has to prepare a case for the board to verify the legal aspects of the OSS adoption or contribution. The board makes a recommendation after case investigation that include IPR matters review. Consequently, the business manager accepts or rejects the original request from the engineers. This decision process may however be very demanding and inefficient in terms of time and resources. To lessen the bureaucracy, Sony Mobile uses frame agreements that can be created for OSS ecosystems that are generally considered as having 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.

4.3 Contribution Strategies based on Artifact Classification

Based on how the software artifacts are classified in the CAP model, different contribution strategies may apply. As illustrated by the four different quadrants in Fig. 3, the artifacts can be classified as one of four different types:

- Strategic Alliances and Investment artifacts
- Platform/leverage artifacts
- Products/bottlenecks artifacts
- Standard artifacts

Below we present and describe different contribution strategies that may apply for each type of artifact classification. To give further context and validation of the different classification types and related strategies, we present examples from Sony Mobile internally but also externally.

Strategic Alliances and Investments Artifacts

This category includes internal and external artifacts that 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. Contributions should hence be regulated and made in a controlled manner. The artifacts should undergo special screening to identify parts that enables the differentiating parts. If possible to modularize and separate, these enabling parts should be selectively revealed [72]. When needed, new OSS ecosystems should be created. In case the artifact is already connected to an existing OSS ecosystem, the firm should strive towards gaining and maintaining a high influence in regards to the specific artifact and attached functionality.

Examples 1 - Gaming, Audio, Video and Camera: Gaming is an area that Sony Corporation considers as a strategic asset and builds its brand proposition on. PlayStation 4 (PS4) console experiences pressure from the online gaming. As a result, game users are more inclined towards streaming games online instead of using consoles. Online streaming requires a large number of virtual machines and Sony Computer Entertainment cannot contribute gaming servers, but the firm can contribute platforms, frameworks, and enablers to facilitate its brand proposition. A typical example of an enabler are 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 to steer the brand proposition for Sony Computer Entertainment since they are needed in order to run the strategic services.

An example of such a framework that Sony Mobile uses is Stagefright[3]. It is used for picture effects of the camera. This framework could be contributed, but not the camera features as these are considered as differentiating towards competition and hence have a high business impact and control complexity for Sony Mobile. I.e., camera effects should not be open, but all enablers should be. Sony Mobile contributes the frameworks to steer and open up a platform for camera experience applications in their mobile phones. A further example of a framework that has been made open but in the context of gaming is the Authoring Tools Framework[4] for PS4.

Platform/Leverage Artifacts

These artifacts include those with a high degree of innovation and positive business impact, but not necessarily needed to be under control of the firm. Examples include technology and market opportunity enablers but that may have competing alternatives available, preferably with a low switching cost. Generally, everything should be contributed, but with priority given to contributions with highest potential to reduce time-to-market. Due to the low need of control firms should strive to contribute to existing projects rather creating new ones, which would require much more effort and resources.

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 in June 2003. DLNA promotes a set of interoperability guidelines for sharing digital media among multimedia devices. The network works with cable, satellite, and telecom service providers to provide link protection on each end of the data transfer process. The layer of Digital Rights Management (DRM) security allows broadcast operators to enable consumers to share their content on multimedia devices without the risk of piracy. Sony Mobile wanted to include a digital network in its handset but there was an OSS solution available in the form DLNA. Therefore, creating a competing solution would not have been wise given that many firms have already chosen the OSS solution. Instead, Sony Mobile also choose to bring the technology in-house since they did not want to invest more than needed. This is a typical example of leveraging functionality that a firm does not create, own, or control, but it is good to have. Therefore, Sony Mobile did not want to commit extra resources to it and use the existing OSS solution to make its offerings better.

Example 2 - Mozilla Firefox: The most significant web browsers during 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

[3] https://github.com/fireworm0/Exploit-Android-Stagefright
[4] https://github.com/SonyWWS/ATF

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 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

This category includes artifacts that does not have a high business impact, but would have a negative effect if not present. For example, functionality required in certain customer-specific solutions but not for the general mass-market. These artifacts are hard to acquire and requires high degree of control due to the specific requirements. Generally, everything should be contributed, but with priority given to contributions with highest potential to reduce time-to-market. 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. An alternative option would be to identify and target specific stakeholders of interest and create a limited project and related ecosystem.

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 smartphones 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 use Microsoft's PlayReady DRM mechanism. This requirement applied to all handset manufacturers, including Sony'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. DoCoMo accepted, which allowed Sony Mobile and its competitors to share maintenance and development on 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

This category includes artifacts that may be considered as commodity to the firm. They do not have a competitive edge if kept internal and has reached a step 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 highest cost reduction potential.

Example 1 - WiFi-connect[5]: This OSS checks whether or not 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[6]: 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.

Both examples are considered standard artifacts because they can be considered as commodity, accessible for competition and does 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 contributing strategies in to practice require appropriate processes and information artifacts in order for developers to know which requirements, or what parts of a them that should be contributed. Analogously, to follow up how contribution strategies are followed, there needs to be a possibility to perform check what has been contributed. In this section we address RQ2 and propose an information meta-model which can be used to record and communicate the operationalization of the CAP model.

5.1 An Information Meta-model in Support of the CAP Model

Software-intensive firms use different repositories to store information about their products as they are planned, developed, released and maintained. By connecting different repositories and their artifacts, information about a specific product may be gathered, traced and measured through its life-cycle and serve as a basis for related decision-making and follow-up processes. For software-intensive firms engaged in OSS ecosystems, it is important to keep track of artifacts that

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

are kept internally and also, what artifacts are contributed to the OSS ecosystem. Consequently, internal and external contribution repositories help firms to analyze the data and adopt the right contribution strategy depending upon the type of the contribution, see section 4.2).

Hence, to identify potential structure of our information meta-model, we investigated available software artifact repositories at Sony Mobile in collaboration with relevant experts. Early investigations suggested that we were able to trace commits and patches that are contributed, via patches, to the specific requirements and feature packages, and finally to the platforms deployed in the Sony Mobile's products, see Fig. 4. The detailed trace with a selection of their respective attributes is mentioned in Table 2. The repositories in question and their connections are set-up according to Fig. 4. The figure shows six separate repositories:

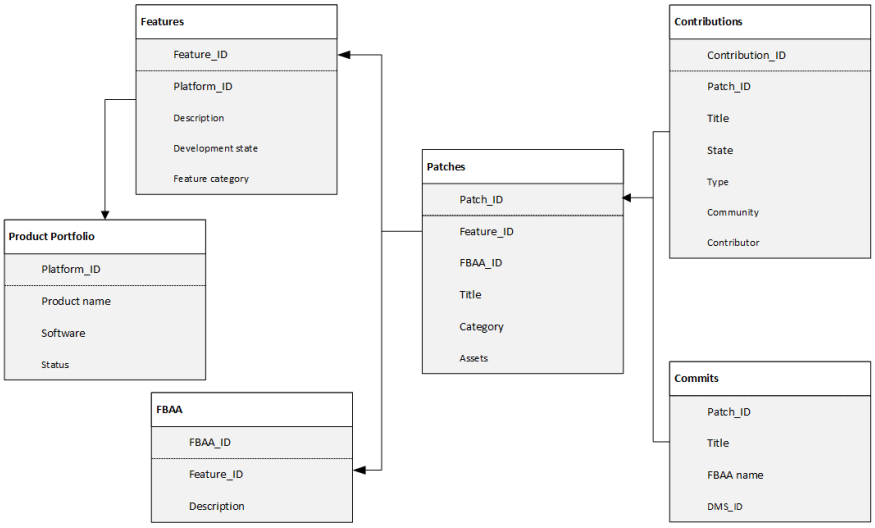


Figure 4: Data repositories necessary to run the CAP model analysis.

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

Table 2: Description of selected attributes from the software artifact repositories mentioned in Fig. 4

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 patches repository.
	Title	A detailed description of a commit.
	FBAA name	Commits associated with the FBAA.

Sony Mobile uses a software platform strategy, where one platform may be deployed in multiple models of their embedded devices. This information is saved in the platform repository where different configurations between platforms, hardware and other major components are stored along with market and customer related information.

Each platform release contains a set of features. The artifacts function as information containers, which can be assigned and updated to different roles as the artifact 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. A special attribute, which is classified based on the decision processes and impact analysis, is the contribution strategy. This attribute is used to classify whether the requirement captured by the artifact may be contributed or not.

Features make up functionality common for multiple platforms, and these common features grouped together and refereed as Feature-Based Architectural Assets (FBAAs). For example, features connected to power functionality may grouped together in its own FBAA and revised with new versions as the underlying features evolve along with new products. By adding needed FBAAs together, different products can be defined.

Even though Sony Mobile uses an OSS project as a base for all of their platforms, customization and new development is needed in order to meet the expectations from the firm's customers. These adaptations are stored as patch artifacts in the patch repository. The patch artifacts contain information about the technical implementation and serves as an abstraction layer for the code commits which are stored in a separate repository. Each patch artifact can be traced to both FBAAs and features.

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

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 spent on the related work. Hence, this set-up makes up a critical part in both the structuring and execution of the CAP model.

5.2 Combining the CAP Model and the Information Meta-model

The CAP model allows Sony Mobile to classify software artifacts as contributable or not based on their business impact and control complexity. The abstraction level of this type of software artifact may range between a complete project, to a

specific component, or a single requirement. 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 criticality and control complexity. In these cases, features included in a certain FBAA would inherit the decision of whether it can be contributed or not. However, pending on the type, size and specification of specific requirement, these may also be suitable for individual classification. This is a choice that is up to the specific firm adapting this way of structuring their contribution strategies.

The tracing structure depicted in Fig. 4 allows for communication of whether an FBA or a feature should be contributed or not, or if a certain contribution strategy should apply in general. In the features repository there is already such an attribute in place, see table 2. This attribute could potentially be introduced in the FBA repository as well, and then inherited to the requirements repository, where it also could be overridden if needed. The communication also goes the other way; by following up on what contributions has been made to each OSS ecosystem, and trace these back to the requirements and FBAAs, the firm can see how the contribution strategies and decisions made are executed. This also offers the possibility to get an overview of how much resources are spent in regards to contributions, and how much is invested in specific OSS ecosystems.

This latter kind of follow-up enables firms to perform a cost value analysis on a OSS ecosystem. The cost is given by the resources invested in the contributions made, although this has to be adapted to also consider other types of interaction with the OSS ecosystems, such as taking part in discussions and support. The value should be determined from case to case but could consider how the OSS project aligns with the internal product road-map and the level of influence that the firm has in the OSS ecosystem. Specific key-performance indicators could include number of proposed requirements that have been included, average resolution time of proposed requirements etc. This kind of analysis could allow the firm to better plan and adapt their resource investment relative to the influence they wish to have in a specific OSS ecosystem. Hence, resources may be spent more efficiently and render in a better return on investment, as is the main motivation behind employing contribution strategies [175].

6 Discussion

In this section, we discuss how the CAP model can be used to elicit contribution strategies. We also discuss its generalizability in relation to similar models and how it should be improved or adapted to fit other contexts.

6.1 Improvements and Adaptation Possibilities of the CAP Model

Influence Needed to Control

The Kraljic's portfolio model was originally used to help firms to procure or source material for their production [93]. 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 manufacturers to control the quality and future development of the material [36]. In 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 it aligned with the firm's internal product strategy. A good example here is the media frameworks for Sony Mobile, see section 4.3.

These artifacts may require special ownership control why a high level of influence in the concerned OSS ecosystems may be warranted to be able to contribute them. As influence in an OSS ecosystem costs in the form of general contributions and activity [35], a possible strategy is to share the artifact with a smaller set of actors with similar agendas, which could include direct competitors [170]. This strategy is still in-line with the meritocracy principle as it increases the potential ecosystem influence via contributions [35]. Sharing artifacts with a limited number of ecosystem actors leaves some degree of control and lowers the maintenance cost via shared ownership [156, 163]. On the other hand, time-to-market for all actors that received the new artifacts is substantially shortened.

For less critical artifacts, 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 4.3 and 4.3. 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 openly 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.

Independent of the case, critical or less critical artifact in regards to control complexity, the firm needs to determine what level of influence they need in the concerned ecosystem. This factor is not covered explicitly by the CAP model but should be considered as an extra dimension or as a separate decision factor in the contribution strategies which are elicited from the CAP model.

Direct and In-direct Use of OSS based SECOs

The second decision factor originating from the Kraljic's model [93] 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.

Examples of artifacts with a high profit, or high business impact, are those related to features that are differential towards competitors and adds significant value to the product and service offerings of the firm [162], e.g., the gaming services for Sony Mobile, see section 4.3. Analogous, artifacts with low profit are those related to commodity features shared among the competitors, e.g., Linux commodity parts, see section 4.3. This reasoning works in cases where the OSS of concern 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 the therefore mainly contribute in the value creation process [4]. This requires the firms to know the customers well enough in order to be able to judge which features or requirements 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 feature's value becomes harder to judge. This is because the feature 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 feature gives them an advantage relative to its competitors.

Moreover, the secondary role of OSS ecosystems often facilitates software engineering process innovations. This, in turn, could render product innovations that can create or increase the business impact of a feature, e.g., if the feature makes the development or delivery of the product to a higher quality or shorter time-to-market respectively [106]. These factors cannot be judged by marketing, but rather by the developers, architects and product owners who are involved on the technical aspects of software development and delivery. In regards 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 a feature's business impact.

Comparing to Other Commoditization Models

Both commoditization models suggested by van der Linden et al. [162] and Bosch [17] consider how an artifact (feature) 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. In the CAP model, the impact of whether an artifact is to be considered differential or commodity is covered by the business impact factor. However, how quickly a feature (or 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.

Relative to the level of commoditization of an artifact, the two previous model consider how the artifact should be developed and shared. Van der Linden et al. [162] suggested to internally keep the differential features 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 regards to the artifact.

The main novelty of the CAP model in relation to the other commoditization models [17,162] is that it supports strategic product planning and occupies a necessary gap between setting product strategies and product planning via feature selection, prioritization and finally release planning [87]. 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 contributing a feature or keeping it internally developed (more patches and longer time-to-market as consequence). Finally, the CAP model is suited for providing direct guidelines for how to position in an OSS ecosystem's governance structure [5] and how to influence it [35].

Reasons for why a firm would wish to contribute may be unique. Thus, the drivers used by Sony Mobile in the CAP model may not be the same for other firms wishing to adopt the same model. When the contribution drivers and the cost structures are identified, they should be aligned with the firm's understanding for how the value is drawn from the OSS ecosystems. This gives firms a view of the *alternative cost* of keeping an artifact closed, and hence improves the understanding of what should be contributed and how the resources should be planned in relation to these contributions.

6.2 OSS Contribution Strategies based on CAP Model

Below we further describe the different contribution strategies that relates to the four types of artifact classifications available in the CAP model as presented in section 4.3.

Enable Differentiation – Control-driven Contributions

This strategy deals with building joint projects using frequent and open exchange of information in order to create long-term relationships. It should be noted that the top management's engagement is required within the firm to focus on the overall value using shared control and competencies. Control can be gained by investing in creating new OSS ecosystems if the existing does not help a firm to make controlled contributions in terms of not sharing the differentiating competencies.

Example 1 in section 4.3 highlights the importance of contributing the multimedia frameworks to Android as an enabler for the music, videos and gaming services.

Platform/Leverage – Maximize General Innovation by Investing in OSS Ecosystems

The features that are not necessarily needed to be mastered by the firm may be a mix of own, OSS, subcontracted or purchased solutions. However, these features are important to the overall business. The key is to reduce the number of solutions and variants, and co-operate with others in order to maximize the technology and innovation development. The focus here should be competence building to follow mainstream technology development, to reduce time-to-market driven contributions and to head way for innovation. This focus can be achieved by investing in existing OSS ecosystems. Examples 1 and 2 shown in section 4.3 highlights leveraging of artifacts by contributing them to OSS ecosystems and avoiding extensive maintenance resource usage.

Products/Bottleneck Artifacts – Look for Standard Solutions or Market Solutions

Operator features is a good example of potential bottlenecks that are market specific. Although, these features may not give any value to a firm, but they have to be included in order for the firm's products to be ranged for a specific market. Otherwise, there is a strong likelihood for a sales miss in a specific market. In order to deal with such bottlenecks, firms should look for standard or market solutions for secure and long-term technology development. It should also be worth considering whether those requirements should be development in-house or outsourced based on the risk and competence.

Firms should consider creating passive OSS ecosystems and push ownership to feature (or other artifact) makers to share the development cost. Example 1 in section 4.3 shows that Symbian failed to deal with the conflicting needs of operators because they could not make it OSS. However, in example 2 Sony released the requested by DoCoMo PlayReady plugin as OSS, enabling external contributions and bug fixes, even from competitors that also provide mobile phones for DOCOMO, e.g. Samsung. Therefore, opening up is a clear win-win for all stakeholders.

Standard Artifacts – Do Not Fork

In relation to standard artifacts, the ultimate goal of the firms is to reduce the maintenance cost by avoiding to fork the OSS. Firms should focus on reducing the number of solutions and variants in order to alleviate the patching and contribution costs. The use of standardized solutions and frequent OSS contributions should be preferred instead of forking or fragmenting, see examples in section 4.3. In

these examples, creating a competing solution could lead to unnecessary internal maintenance costs, which has no potential of triggering a positive business impact for a firm.

7 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. At the same time, the acceleration benefits require improved support for strategic product planning in terms of clear guidelines of *what to develop internally* and *what to base on OSS*. Currently available commoditization models [17, 162] accurately capture the inevitability of commoditization 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) developed to bridge business models and product strategy with operational product planning and feature definition. 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 allows 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. CAP is an important step for firms that use these ecosystems in their product development and want to increase their OI benefits. 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.

In relation to *RQ1*, we propose the CAP model to classify artifacts based on control complexity and business impact using four main motivators 1) cost 2) time-to-market (TTM) 3) control and 4) strategic alliances and investments. In relation to *RQ2*, we present the instantiation of the CAP model in terms of an information meta-model that could be used by software-intensive firms working with OSS ecosystems to keep track on the contribution strategy realization and feature management. This brings the potential for managers to better understand the OI impact and benefits for the software products offering.

In future work, we aim to validate the CAP model and related information meta-model in other firms. We plan to focus on understanding the firm specific

and independent parts of the CAP model. At the same time, we plan to continue to capture operational data from Sony Mobile related to the usage of the CAP model that will help in future improvements and adjustments.

CHAPTER V

A STAKEHOLDER ANALYSIS FRAMEWORK FOR OPEN SOURCE SOFTWARE ECOSYSTEMS

Johan Linåker, Björn Regnell, Daniela Damian

Abstract

Background: Software-intensive firms involved in Open Source Software (OSS) ecosystems are part of an internal and an external Requirements Engineering (RE) process. The latter regards the informal RE process of the OSS ecosystem where the focal firm is one in a larger set of stakeholders who collaborate on realizing the requirements in the OSS. For a focal firm to impose its own agenda, they must consider the agendas of other stakeholders and the influence they possess. **Aim:** This study aims to enable firms to identify and characterize stakeholders in OSS ecosystems in terms of their influence and interactions to create awareness of conflicting agendas and understanding of how to respond by building and leveraging an influence on the ecosystem's RE process. **Method:** By applying a design science approach, this study proposes a framework based on literature in three conceptual foundations. **Results:** The framework is adapted to consider the special characteristics of OSS RE and enables firms to structure their stakeholder identification and analysis processes toward OSS ecosystems. Social network constructs and processes are used to measure the influence and interactions of the stakeholders. The framework is validated analytically and descriptively through a case study on the Apache Hadoop OSS ecosystem. **Conclusions:** The framework adds a strategic aspect of stakeholder identification and analysis by specifically addressing the attributes of influence and interactions, both essential in the informal, collaborative and often meritocratic RE processes and culture of OSS ecosystems. In future

work, we aim to refine and validate the framework through further design cycles with expert opinions and case applications.

1 Introduction

Open Source Software (OSS) has proved to play a pivotal part in many software-intensive firms' product strategies and business models [26, 127], e.g., as a basis for support and professional services, as an open core or platform for proprietary extensions, as part of a dual-licensing model, or as a way to create a third-party ecosystem. These different ways of revealing and exploiting internal and external code and knowledge allow OSS to be leveraged as a means to advance internal innovation and technology capabilities (cf. the definition of Open Innovation (OI) [25]). For the firm, usage of OSS in this context of OI often implies a membership in an ecosystem of other stakeholders who are also exploiting the OSS from the perspective of their individual incentives [80]. With stakeholder we refer to Glinz & Wieringa's definition: A *"A stakeholder is a person or organization who influences a system's requirements or who is impacted by that system"* [61]. In our context, we consider person or organization as the members of an OSS ecosystem, and system being the OSS that underpins the ecosystem [80]. The ecosystem membership causes the firm's borders to become permeable for interaction and influence from the ecosystem's continuously evolving population of both known and unknown stakeholders [127].

This fluctuating population may imply several challenges [107, 127], including conflicting agendas, hardship to align internal strategies with the OSS ecosystem, and difficulty in constructing contribution strategies. The latter is essentially related to requirements scoping activities that helps firms to maintain a competitive edge by offering guidance on what software artifacts to contribute, how and when [175]. Giving away differentiating intellectual property, especially to competitors, may have detrimental effects on both for existing and future business [162]. This highlights the importance of stakeholder identification and analysis to provide strategic input for firms involved in OSS ecosystems. From a general stakeholder theory perspective [55], this input should address three questions;

Q1 Who are they? - Regards the characteristics and attributes of the stakeholders.

Q2 What do they want? - Regards how the agenda of the stakeholders align with that of the focal firm.

Q3 How are they going to get there? - Regards what strategy the stakeholders use to satisfy their agendas.

In order to answer the latter two, firms must first address the former (**Q1**). One of the more important attributes in this regard is that of power [118]. However, due to the informal and collaborative nature of the Requirements Engineering

(RE) process [149] and the often meritocratic governance structure in OSS ecosystems [128], this term translates better as influence. The Merriam-Webster dictionary [1] defines influence as *"the power to change or affect someone or something"*. In our context this translates to the power of a stakeholder to change or affect the RE process in an OSS ecosystem. This notion of influence aligns naturally with what defines a stakeholder [61], and enables firms to see the requirements in which stakeholders hold a certain interest, and from there be able to create an overview of their agenda in the ecosystem (**Q2**). Further, this understanding also enables the focal firm to analyze how these stakeholders invest their resources in order to satisfy their agendas (**Q3**). By also considering with whom stakeholder interacts and how, firms may identify possible partners and competitors, but also to learn how to adapt their own strategies and processes with the OSS ecosystem's. This creates further understanding for how firms can build their own influence, and leverage it towards other stakeholders in the ecosystem's RE process.

Existing stakeholder identification practices are not adapted to specifically consider the aspect of stakeholder influence in the context of RE in OSS ecosystems [136]. We address this gap with a framework that enable firms to identify and analyze an OSS ecosystem's stakeholders, and study their influence by the impact they have with respect to the requirements that get implemented in the OSS. With a design science approach [73], we base our framework on social network analysis constructs [165] that has proven use in characterizing the influence of stakeholders [145], but also when analyzing firm's participation in OSS ecosystems [135]. The framework adopts an analysis approach used in earlier work [108] and formalizes it to further consider the informal and decentralized RE processes present in OSS ecosystems [149]. We validate it through a case study (partly based on earlier work [108]) of the Apache Hadoop OSS ecosystem. First, we analytically [73] investigate how the social network analysis constructs proposed by literature to measure influence (e.g., [46, 131, 135, 145]) associate with two performance measures which are defined to reflect a positive performance for a firm in regards to an OSS ecosystem's RE process. Second, we validate the framework descriptively [73] by applying it on the stakeholder population of the Apache Hadoop OSS ecosystem.

The rest of this paper is dispositioned as follows: In section 2 we describe the research approach used in the development of our framework. In section 3 we present our framework and its conceptual foundations, while in section 4 we validate and apply it through a case study. In section 5 we discuss the framework and its validation, followed by a discussion of threats to validity in section 6. Finally, we conclude the paper in section 7

[1] http://www.merriam-webster.com/dictionary/influence

2 Research Approach

The framework proposed in this study was created by using a design science approach inspired by Hevner et al. [73]. Hevner et al. proposes seven guidelines for researchers to consider when conducting design-science research, these are: *(1) problem relevance, (2) research rigor, (3) design as a search process, (4) design as an artifact, (5) design evaluation, (6) research contribution,* and *(7) research communication*.

The problem context (1) of the study is the open and fluctuating stakeholder population in OSS ecosystems [107,127,175], which for involved firms may imply conflicting agendas, hardship to align internal strategies and RE processes with the ecosystem, difficulty in constructing contribution strategies, and finding a sustainable and profitable position in the ecosystem's governance structure. To address this problem, we aim to develop a technology-based artifact in the form a framework to help firms situated in the problem context (4) to systematically structure their analysis process of stakeholders in OSS ecosystems. In the development, we use an iterative process through design cycles (3). One cycle consists of a two-step process of first building a prototype of the artifact, and then evaluating it (5) through different validation-steps [73]. The framework is constructed and presented with the intention to be applicable and useful from a practitioner's point-of-view (6 and 7).

In section 3 we describe the framework and its conceptual foundations, but also motivate its construction. In section 4 we validate the framework by applying it in a case study on the Apache Hadoop OSS ecosystem (partly based on earlier work [108]). First, we analytically [73] investigate how the social network analysis constructs proposed by literature to measure influence(e.g., [46, 131, 135, 145]) associate with two performance measures which are defined to reflect a positive performance for a firm in regards to an OSS ecosystem's RE process. Second, we validate the framework descriptively [73] by applying it on the stakeholder population of the Apache Hadoop OSS ecosystem.

3 Stakeholder Analysis Framework for Open Source Software Ecosystems

The framework aims to enable firms involved in OSS ecosystems to structure their stakeholder analysis process systematically as part of their RE process towards the ecosystems, see table 1. Focus is specifically on identifying and characterizing stakeholders' interactions and influence on the RE process in the OSS ecosystem. First we describe the background and conceptual foundations underpinning the framework, and then move on to give a detailed overview of the frameworks different parts.

Table 1: Overview of the framework and its six sequential steps (S1-S6) along with related descriptions and examples.

	Step	Description
S1	Determine the purpose of the analysis process.	Purpose could include: • Identify potential partners or competitors overall or in regards to a certain feature. • Identify influential stakeholders learn from in order to raise one's own influence in the OSS ecosystem. • Identify stakeholders with conflicting agendas in regard to a certain feature. • Identify temporal collaboration patterns through releases.
S2	Limit the scope based on analysis purpose.	Regards boundaries for what data that should be collected and is determined by the purpose of the analysis process. E.g., is the interest limited to • a certain set of component or feature of the OSS? • a certain set of, or individual stakeholders? • a certain time-period or releases?
S3	Identification and mining of requirements artifact repositories.	Refers to the main repositories through which stakeholders interact in regards to the RE process. E.g., • IRC or other chat-based communication • Issue trackers • Code review • Software code repository • Discussion boards
S4	Classification of organizational affiliation.	Concerns identification of organizations to which individual developers are affiliated. E.g., by • Interacting and studying the communication within an OSS ecosystem. • E-mail domain analysis. • Heuristically through social media and public electronic sources. • Identity pattern matching.
S5	Creation of stakeholder interaction networks.	For each requirement artifact repository, a directed and weighted affiliation-network is created. Stakeholders are represent as nodes, and are connected by edges if they have interacted on a common requirements artifact, e.g., commented on the same issue or mail-thread. To reflect investment and influence, edges are weighted in order to reflect the size of each stakeholder's participation.
S6	Influence analysis of stakeholder interaction networks.	To characterize stakeholders' influence on the RE process in the OSS ecosystem, a set of network centrality measures are applied to the interaction networks created in **S5**. These include: • In- and Out-degree centrality • Betweeness centrality • Closeness centrality • Eigenvector centrality

3.1 Background and Conceptual Foundations

The framework's construction is based on three conceptual foundations.

- Building Influence on an Informal and Collaborative Requirements Engineering Process.
- Leveraging Awareness of Dynamics behind Stakeholder Influence and Interrelationships.
- Using Social Network Constructs to Characterize Stakeholder Interactions and Influence.

Below we describe and present these one by one.

Building Influence on an Informal and Collaborative Requirements Engineering Process

Unlike traditional bespoke or market-driven RE [142], RE practices in OSS ecosystem may be described as informal and decentralized. There is no central repository with requirements defined in the problem space, describing the product of need, along with heavy processes and tools for examining the requirements for completeness and consistency [3]. Instead, RE may be considered as a lightweight and evolutionary process of requirements refinement [45]. Practices such as elicitation, specification, and validation overlap and are done collaboratively through iterative and transparent discussions including up-front implementations [45, 60, 149]. These discussions and implementations of requirements are spread out over a multiple number of requirements artifacts, each with its own repository. Examples of these artifacts (cf. informalisms [149]) include reports in an issue tracker, messages in a mailing list, or commits in a version control system.

Prioritization is commonly conducted by the core-team overseeing the project management, though care is often taken to the opinions of other developers and users [97]. Core-team members and others with a high position in an OSS ecosystems governance structure [5] often attain this influence on the RE process by being active, contributing back, and having a symbiotic relationship with the OSS ecosystem [35]. This governance structure is often referred to as a meritocracy [83]. Nakakoji et al. [128] illustrates this with an Onion model where the outer layer is constituted by the passive user, and the center by the project leader. For each layer towards the center, influence in the ecosystem increases.

Leveraging Awareness of Dynamics behind Stakeholders' Influence and Interrelationships

Besides having a symbiotic relationship with the OSS ecosystem [35], firms need to stay aware of the ecosystem's evolving and dynamic stakeholder population in

order to continue to build, maintain and leverage its influence on the RE process in an OSS ecosystem [127]. Otherwise, the fluctuating population may unknowingly introduce conflicting agendas, hardship to align internal strategies with the ecosystem, and difficulty in constructing contribution strategies [175]. Depending on the focal firm's position and role in the ecosystem's governance structure [5], it may be that the focal firm is no longer the vantage point and instead to be considered a stakeholder among others to the OSS. Hence, firms need to consider the influence of other stakeholders in order to respond to potential threats towards their own agenda and competitive edge [53]. This highlights strategic aspects of continuously identifying and analyzing properties such as interactions and influence in a stakeholder population of an OSS ecosystem, which is not covered by existing methodologies [136].

These properties align with the importance of characterizing stakeholder power as suggested in general stakeholder theory [118]. This characterization is further needed in order to answer first of the three questions that Frooman [55] stipulates in regards to what needs to be addressed in a stakeholder analysis: who they are in regards to their attributes, what they want in terms of their agenda, and how they will get there in terms of achieving their agenda. By identifying and characterizing the stakeholders' influence in on the RE process, firms are enabled to see in what requirements the stakeholders hold a certain interest, and from there be able to create an overview of their agenda in the ecosystem [55]. Further, this understanding also enables the firms to analyze how these stakeholders invest their resources in order to satisfy their agendas. By also considering with whom stakeholder interacts and how, firms may identify possible partners and competitors, but also to learn how to adapt their own strategies and processes with the OSS ecosystem's. This creates further understanding for how firms can build their own influence, and leverage it towards other stakeholders in the ecosystem's RE process.

Using Social Network Constructs to Characterize Stakeholders' Interactions and Influence

To characterize a stakeholder's influence on the RE process in an OSS ecosystem, social network constructs may be used to quantify stakeholders' position and prominence relative each other [145]. A stakeholder is more prominent if it has a central position in the network with edges that make it extra visible and important to others [7]. In social networks, centrality measures are commonly used to analyze an actor's position and prominence relative others [165]. Faust [46] breaks down the notion of centrality and explains how an actor is central given that they are active in the network, can communicate with others in the network efficiently, are able to mediate and control flow of information between others in the network, and have relationships with others that are central. These four aspects respectively relate to the centrality measures of degree, betweeness, closeness and eigenvector

centrality. We use these measures as the foundation for analyzing the influence of stakeholders in our framework.

Social networks are commonly used to characterize relationships and centrality in RE [136]. Damian et al. [38] construct what they refer to as requirement-centric social networks which visualizes the social structure and relationships among individuals in development teams working with a certain requirement. With the use of different social network measures, collaboration around the requirement may be broken down and studied, e.g., through the aspects of communication, coordination and awareness. Important nodes to look for in a network are what Marczak et al. [115] refer to as information-brokers. These bridge two or more subgroups and make up critical parts in the coordination and information flow of a network. In another study, Bhowmik et al. [10] show that stakeholders in this position are more probable generate a higher number of new requirements in comparison to the rest of the networks. Lim et al. [104] identifies stakeholders and their roles by letting stakeholders recommend others.

In OSS ecosystems, many studies have focused on a developer and user level (e.g., [12, 31, 41], while those focused on an organizational level are fewer, with some exceptions. For example, Orucevic-Alagic et al. [135] investigated the influence and collaboration of stakeholders on each other in the Android ecosystem. Texiera et al. [158] explored collaboration between firms in the Openstack ecosystem from a co-opetition perspective showing how firms, despite being competitors, may still collaborate within an ecosystem. A similar study was also performed on the Webkit ecosystem [157]. Linåker et al. [108] investigated the interaction and collaboration patterns between firms as stakeholders in the Apache Hadoop ecosystem by looking at their patch-contributions to specific issues using a similar approach as Orucevic-Alagic et al. [135]. A qualitative investigation similar to that of Texiera et al. [158] was also performed confirming the observation of co-opetiton.

3.2 Summary

Stakeholder identification and analysis play a pivotal role in regard to help firms answer questions as what stakeholders are present, what their agendas are, and how they aim to achieve them [55]. However, current practices [136] are not adapted to consider these strategic aspects [53] implied by the membership of an OSS ecosystem [127] and its informal and collaborative RE process [45, 149]. Involved firms are no longer the vantage point, and instead part of a larger set of interdependent stakeholders [145]. This study contributes by developing a framework to help firms in this context to systematically structure their analysis process of stakeholders in OSS ecosystems. As proposed by earlier work [55, 118, 145], the framework focus on analyzing the attribute of influence of stakeholders, and specifically in regards to the RE process of the OSS ecosystem, by applying social network constructs in line with earlier work [108, 135, 145]

3.3 Detailed Description of Framework

Below we describe the framework in its five steps as presented in table 1.

Determine the Purpose of the Analysis Process (S1)

The framework offers a systematic approach to how a firm may structure its analysis process, it does however not describe how to apply or use it. It is the purpose for behind the analysis that should determine this. The purpose is further needed to guide the structuring of the analysis process. Hence, the first step (**S1**) of the framework is to determine what questions are of interest to answer based on the stakeholder analysis. E.g., to identify potential partnerships or competitors, to identify and learn from stakeholders in a certain position, or identify conflicting agendas in regards to certain requirements.

Limit the scope based on analysis purpose (S2)

Based on the purpose of the analysis process, limitations may be implied that can affect how the analysis should be narrowed down in terms of what requirements artifacts should be included in the analysis. Questions to be asked include if the interest limited to

- a certain set of component or feature of the OSS?
- a certain set of, or individual stakeholders?
- a certain time-period or releases?

Identification and Mining of Requirements Artifact Repositories (S3)

As highlighted in section 3.1, requirements in OSS ecosystems can be represented by multiple artifacts (cf. informalisms [149]) in different types of repositories. These artifacts capture and persist the interactions between the stakeholders in regards to the underlying requirement and related RE sub-processes (e.g., elicitation, specification, analysis and prioritization [96]). Hence, third step in the analysis process suggested in our framework (**S3**) is *to identify what types of repositories that are mainly used by the OSS ecosystem.* Examples include issue trackers, mailing-lists, IRC logs, source code repositories, and code-reviews. When these are identified, the repositories should be mined to collect the necessary data. This can either be done either by existing [2] or custom-made tools.

[2] http://metricsgrimoire.github.io/Bicho/

Classification of Organizational Affiliation (S4)

When relevant requirements artifact repositories has been identified and requirements artifact gathered based on the purpose of the analysis, the *individuals that are involved in OSS ecosystem needs to be classified in regards to affiliation* (**S4**). This is a necessary step as firm-affiliated individuals may be assumed to represent the agenda of their sponsor or employer. However, not all individuals involved in an OSS ecosystem have to be affiliated and may rather represent their own personal agenda. These affiliations can be identified and triangulated by qualitative and quantitative means. E.g., through involvement and discussions, and by analyzing meta-data from the requirements artifact repositories and cross-checking against other information sources (e.g., social media and electronic archives). Similar triangular and heuristic approaches have been used in previous research [13,62,108].

Creation of Stakeholder Interaction Networks (S5)

When affiliations have been determined, *interaction network for each requirements artifact repository needs to be created* (**S5**) in order to visualize the interactions between stakeholders. From a social network perspective, a requirements artifact may be considered as an event which involves the relevant RE sub-processes surrounding the underlying requirement. Stakeholders that have interacted through these RE sub-processes can be said to be participants of the same event. These events and their participants can be represented by networks where the actors represent the stakeholders, and the edges that connect the stakeholders represents the interaction between them that has taken place through the event (cf. requirement-central networks [38]). To analyze a specific requirement, a separate network can be created for each related requirements artifact and used in conjunction to understand what stakeholders are involved and how they interact. In a similar fashion, sets of requirements (e.g., a feature, module or complete project) may be analyzed by aggregating requirements artifacts in a repository to network. The different networks should still be analyzed in conjunction to get complete overview of what stakeholders that are involved and how they interact.

It should be noted that one stakeholder's participation in the event representing a requirements artifact and it RE sub-processes may be of a relatively different size than the other stakeholders'. A stakeholder with a higher degree of participation may be considered to have a larger investment and interest in the event. These differences in investment of time and resources need to be considered in order to give a fair view of a stakeholder's stake in a requirement. The relative size of the investment also helps to give a fairer data-set when doing an influence analysis of the interaction networks. As suggested by Orucevic-Alagic et al. [135], weights can be calculated to describe the relative size of the participation to an event. For example, when creating an interaction network based on an issue-tracker, each issue represents a requirements artifact and posted comments may represent the participation of stakeholders. Given that three stakeholders (A, B and C) comment

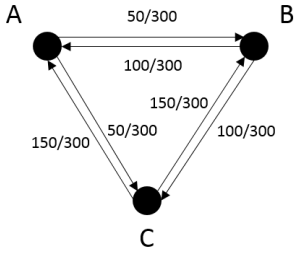


Figure 1: Example of network with three actors A, B and C, and weighted edges connecting them.

on the issue, they are all considered as actors in a network with edges connecting them. To consider the size of their participation, one option could be to use the relative number of comments of each stakeholder. Say A commented 1, B commented 2, and C commented 3 times. This results in the edge weights: A →B&C = 1/5, B→A&C = 2/5, and C→A&B = 3/5.

If two stakeholder participated equal number of times, size of each participation can be made further fine-grained. In another example, when considering an interaction network based on patches submitted to a software code repository, the relative size of a stakeholder's participation can be quantified with the number of changed lines of code (LOC) per patch. A simplified example is shown in Fig. 1 where three stakeholders A, B, and C each created a patch that was submitted. A's patch contains 50 LOC. B's patch contains 100 LOC, while C's patch contains 150 LOC. In total, 300 LOC were contributed to the issue. Resulting in the following edge weights: A→B&C = 50/300, B→A&C = 100/300, and C→A&B = 150/300. Hence, all pairs of stakeholders that have interacted has two edges connecting them pointing from one stakeholder to the other, and conversely in the opposite direction.

By constructing this kind of networks (cf. weighted and directed affiliation-networks [46, 165]), stakeholders' interaction in an OSS ecosystem's RE process may be visualized on different abstraction levels across the different requirements artifact repositories identified in **S3**.

Influence Analysis of Stakeholder Interaction Networks (S6)

In our framework, we are interested to analyze how influential stakeholders are relative each other based on the interaction networks that are created in **S5**. To do this, we leverage the four centrality measures suggested by Faust [46]: degree, betweeness, closeness and eigenvector centrality.

These four centrality measures can be adapted in different ways to provide further facets of influence in regards to the interaction networks. As the interaction

networks are described in **S5**, the edges that connect two stakeholder have weights attached to them. These weights allow the measures to take account for the relative size of each stakeholder's participation of the requirements artifacts on which the network is based on. E.g., out-degree centrality (a sub-set of degree centrality, see table 2) refers to the sum of weights attached to outgoing edges from the focal stakeholder and its adjacent stakeholders [8]. This give an overall number in regards to the size the focal stakeholder's participation in the set of requirements artifacts covered by the network. A high out-degree centrality may indicate that the focal stakeholder has a high influence on its adjacent neighbors and is good at communicating its views relative others in the network [135]. However, this way of measuring out-degree centrality does not communicate the total number of connections of a stakeholder which may better show the number of collaborations and opportunities to spread one's opinions [134]. Hence, we recommend that the proposed centrality measures are used both in the case where the edges have the relative weights attached to them, and in the case where they are considered either present or not [52]. These different types can also be weighted together and correlated against performance measures in order to find the best combination, as proposed by Opsahl et al. [134]. This was tested in the case study of Apache Hadoop but it was found that the pure measures provided the best correlation values.

In table 2, we describe the foundation for these measures and how they may be interpreted generally. In table 3, we give our interpretation of these measures in terms of a stakeholder's influence on the RE process of an OSS ecosystem.

As described by Faust [46], centrality may be broken down in to multiple aspects. Centrality measures in turn uses different definitions and sets of criteria in regards to what classifies an actor's position as central. Hence, each measure can present a different social structure than the other and provide their own perspective on who are the most active [135]. In smaller and simpler network structures such measures may co-vary, while in larger and more complex networks, they may characterize actors very differently [65]. Therefore, measures presented in table 3 should be seen as complementary each other and used together when analyzing a stakeholder activity network. Brass [19] for example, describe how betweeness and closeness centrality each corresponds two the two aspects of acquiring power, decreasing one's dependence (resulting in higher betweeness centrality) and increasing one's independence (resulting in higher closeness centrality.

4 Application of Framework: Case Study of Apache Hadoop Ecosystem

In this section, we seek to in a first step validate our framework descriptively [73] through a case study [147] on the Apache Hadoop ecosystem. First, we give a general introduction to the Apache Hadoop ecosystem. We then provide two examples of how the framework can be applied on the ecosystem. Lastly, we investigate

Table 2: Network measures described from a general perspective as well as and how they can be interpreted in a general sense.

Measure	Description	General Interpretation
Degree Centrality	This measure actually consists of two measures: *In-degree* and *Out-degree centrality*. The former refers to directed edges where the focal actor is the receiver (target) of the edges, while the latter refers to the opposite, when the focal actor is the transmitter (source) for the edges. With weights considered, these measures refer to the sum of weights attached to incoming or outgoing edges of the focal actor [8]. With binary edges considered, these measures refer to the number of incoming or outgoing edges between the focal actor and its adjacent actors [52].	Degree centrality is generally considered as a measure of activity that can identify "where the action is" and highlight the most visible actors in the network [165]. More edges indicate more opportunities in terms of spreading information and exploiting resources. With weights considered, a high in-degree centrality may indicate that the adjacent neighbors have a high influence of the focal actor's work [135]. In a corresponding manner, a high out-degree centrality may indicate that the focal actor has a high influence on its adjacent neighbors and is good at communicating its views to others.
Betweeness Centrality	Refers to the extent to which the focal actor lies on the shortest path between pairs of other actors. With weighted edges considered, it refers to the shortest path with lowest sum of weights [18, 130]. With binary edges considered, it refers to the shortest path in regards to least number of edges [52].	Betweeness centrality is a measure of control and coordination as it highlights actors who sit on the shortest, and sometimes only, communication paths or resource flow between many others [165]. Hence, actors with a high betweeness centrality are potentially able to decide on and influence what information, instructions and resources to pass on to others, but with the option to slow it down, distort and spread based on own best interest. When an actor is the only one, or one of very few, linking two or more parts of a network, they are commonly referred to as brokers as their possibility to influence is very high [131].
Closeness Centrality	Refers to the inverse of the sum of shortest paths from the focal actor to all others in the network. With weighted edges considered, it refers to the shortest path with lowest sum of weights [18, 130]. With binary edges considered, it refers to the shortest path in regards to least number of edges [52]. This measure only considers those actors that are connected to the same network as the focal actor [131]. For disconnected actors, the measure in undefined as the distance is infinite.	Closeness centrality is a measure of efficiency in contacting others and spreading, but also receiving, information in the network and hence an actors' ability to influence others [131]. Actors with a high centrality can be considered more independent as they are in need of fewer intermediaries that may risk delay and distort the information, or spread it in unfavorable directions [145].
Eigenvector Centrality	Refers to how connected an actor is, similar to degree centrality, but considers how well-connected the adjacent actors are [16]. The focal actor receives a score based on a sum of its adjacent actors' scores [131].	Eigenvector centrality is a measure of activity and visibility as degree centrality, but adds information to whom these attributes connect to. A high value indicates that the actor has important friends who in turn are visible and active [131]. Hence, the focal actor is in a position to better communicate and influence key actors in the social network [46].

Table 3: Network measures described how they can be interpreted from an RE perspective in regards to Stakeholder influence.

Measure	Stakeholder Influence interpretation
In-degree Centrality	With weighted edges, a high in-degree centrality is an indication of influence from the adjacent stakeholders' relative the focal stakeholder as they have contributed to a large part in the requirement artifacts which they have interacted with. However, with binary edges, a higher in-degree can be an indication of influence from the focal stakeholder on those adjacent. The focal stakeholder may be considered popular as others seek its opinions.
Out-degree Centrality	With weighted edges, a high out-degree centrality is an indication of influence on adjacent stakeholders as the focal stakeholder has participated to a large part in the requirement artifacts which they have interacted with. This participation can be viewed as the focal stakeholder's opinions in the RE process of the OSS ecosystem. In both cases of weighted and binary edges, a higher out-degree may also indicate a higher number of options or opportunities for qualitative contacts, i.e., to know the key stakeholders to influence and create traction with on a certain issue. For binary edges specifically, it may further indicate a high level of activity through number of collaborations, but also to which the focal stakeholder has expressed its opinions.
Betweeness Centrality	A high betweeness centrality indicates that the focal stakeholder may control and coordinate the information flow about requirements, and interactions between other stakeholders. The focal stakeholder could be characterized as having a central position in the ecosystem, e.g., in regards to project management and governance. Others may be dependent on the focal stakeholders to relay the information and to set-up connections. Further, the centrality also indicates the ability to act as an intermediary that can influence the content of the information, and who it reaches and when, to better serve personal priorities.
Closeness Centrality	A high closeness centrality indicates that a stakeholder is efficient in spreading and receiving information about a requirement to and from the rest of the network of stakeholders. This efficiency allows the focal stakeholder to more easily communicate its agenda on the requirement and interact with others, e.g., in negotiations and lobbying. The focal stakeholder could therefore be characterized as being close to other stakeholders and more independent. This further minimizes the risk of intermediaries influencing the information about the requirement in an unfavorable manner.
Eigenvector Centrality	A high eigenvector centrality indicates that a stakeholder knows and collaborates with other stakeholders who are important and have key positions in the OSS ecosystem. The focal stakeholder is in a position to have a potentially high impact on the RE process in the ecosystem by being able to communicate its agenda to, and influence key stakeholders.

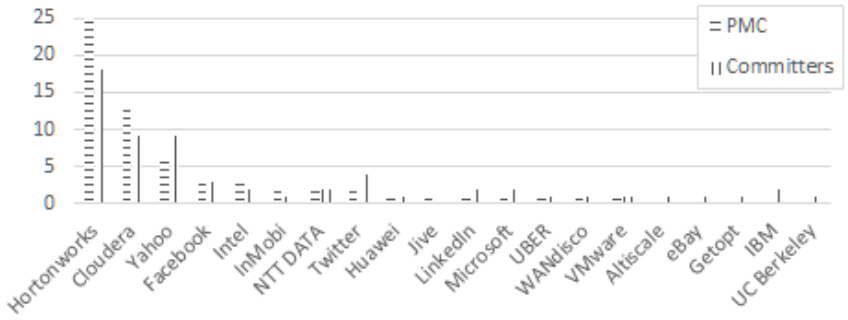


Figure 2: Number of committers and members in the Apache Hadoop PMC aggregated per firm.

if we can find an association between the measures proposed in the framework (see table 2) and performance measures that can represent actual influence on the ecosystem's RE process.

4.1 About Apache Hadoop Ecosystem

The Apache Hadoop project [3] is a widely adopted OSS framework for distribution and process parallelization of large data. In this case study we are specifically interested in the ecosystem surrounding the project, which may be viewed as the technological platform underpinning the relationships between the actors of the Apache Hadoop ecosystem [82]. We focus our scope inside of the ecosystem, its actors and the relationships between them [80].

The Apache Hadoop ecosystem is governed by a Program Management Committee (PMC) that consists of representatives from the Apache Software Foundation and of elected members from the project's ecosystem. Further, the PMC members are also classified as committers, i.e., they have been granted write access to the project. A member may be elected as a new committer by the existing ones. Being elected as a committer does however not imply a membership of the PMC. To become a committer or member of the PMC, an individual need to show merit, e.g., by contributing and actively participating the development of the project. Hence, power may be earned by showing a long-term commitment and having the competence needed. In Fig. 2 the distribution of members among the committers and the PMC are presented based on affiliation per firm.

[3] http://hadoop.apache.org/

4.2 Potential Application Examples

Here we present two cases for how the framework may be applied the Apache Hadoop ecosystem to analyze: (1) the ecosystem's overall landscape of stakeholder interaction and influence and (2) a particular stakeholder's influence and area of interest within the ecosystem. We will be addressing the steps as presented in table 1.

Overall Landscape of Stakeholder Interaction and Influence

In this example, we are interested in getting a general overview of the Apache Hadoop OSS ecosystem (**S1**). We want to see who are the most influential stakeholders, both in regards to social and technical interaction over a certain time period. This implies that we will limit the investigation to examining requirements from a certain release-set, but will consider all involved stakeholders (**S2**). More explicitly, we limit our scope to requirements included from release 2.2.0 (15/Oct/13) to 2.7.1 (06/Jul/15). Further, we are interested in looking at a social and technical interaction in regards to choice of requirements artifacts. This renders in the choice of the issue-tracker in regards to requirements repository (**S3**). This repository may be used to represent both the social and technical interactions as issues contain both comments and patches. The patches are committed by authorized authorized users, once they have been approved. To identify the organizational affiliation of individuals that have interacted via the requirements (**S4**), we perform an analysis of e-mail subdomains, complemented with cross-checking against other information sources (e.g., social media and electronic archives) [13, 62, 108].

Based on the scope specified in **S2**, and the repository identified in **S3**, two interaction networks were generated: a *comments-network* based on stakeholders who have commented on common issues, and a *patch-network* based on stakeholders who have contributed patches to the same issues (**S5**). The patch-network has been presented in earlier work [108], and a similar data collection and cleaning approach was used in order to create the comments-network, as is also proposed in the framework (see section 3). The comments-network shows activity and collaboration of a stakeholder in regards to the social interaction and discussion that revolves around a certain issue, and the patch-network shows same characteristics for a stakeholder in regards to suggesting technical implementations.

In each of the two networks, a stakeholder is represented by a node, and the collaborations between them are represented by the edges connecting the nodes. The comments-network consists of 122 stakeholders, compared to 86 stakeholders in the patch-network (see table 4). In both cases, this includes two groups of developers classified as independent or as unidentified. The comments-network has a higher degree of collaboration with an average of 9.0 collaborations per stakeholder, compared to the patch-network, which has an average of 3.0 collaborations per stakeholder. Both networks are visualized on a high level in Fig. 3 and 4. La-

bels are of firms and of relative size to their weighted out-degree, why only a those with the highest values may be readable.

Table 4: Characteristics of the comments- and patch-networks.

	Comments-network	Patch-network
Stakeholders	122	86
Collaborations	1096	260

To measure the influence of, and collaboration among, the stakeholders (**S6**), three SNA measures were leveraged: weighted out-degree, betweeness and closeness centrality. Other centrality measures presented in table 2 were excluded due to space reasons. In Fig. 5- 7, the three measures are presented in one diagram each. The diagrams contrast the respective measures for the comments- and patch-networks in regards to the 30 top stakeholders. The measures measure different aspects of influence and collaboration among the stakeholders. Below, the three measures are compared and contrasted in regards to the two networks.

Fig. 5 illustrates the normalized out-degree centrality which may be considered rather equal for most stakeholders with the exception of those most influential: NTT Data, Yahoo, Hortonworks and Cloudera. Both NTT Data and Yahoo have a notably higher influence in regards to technical implementation-suggestions, while Hortonworks and Cloudera have a higher influence and activity through social interaction and discussion. Considering the distribution of stakeholders from the different user categories, a heavier representation of product vendors (Hortonworks, Cloudera and Huawei) can be seen in the top five, in regards to both the comments- and patch-networks.

In Fig. 6, it can be seen that the normalized betweeness centrality varies notably between the comments- and patch-networks for the top stakeholders. Hortonworks has the highest betweeness centrality in regards to both the technical and social aspects, and compared to Cloudera and Yahoo, it has double the betweeness centrality in the comments- and patch-networks respectively. Contrasting Cloudera and Yahoo, a clear difference in focus and importance is shown. Cloudera values technical implementation suggestions over social interaction and discussion, while Yahoo focuses on social interaction and discussions. Comparing the two networks overall, the patch-network only has 15 stakeholders with a betweeness centrality higher than zero, while the comments-network has the double amount. This aligns with the earlier observations of the comments-network as more connected and more collaborative than the patch-network.

A high closeness centrality indicates a stakeholder's relative facility in spreading information to, and receiving information from, the network. Fig. 7 visualizes the normalized closeness centrality of the comments-network and the product vendors Hortonworks, Cloudera and Huawei in top, while NTT Data, Yahoo and Hortonworks top the patch-network, closely followed by Huawei and Baidu. Some

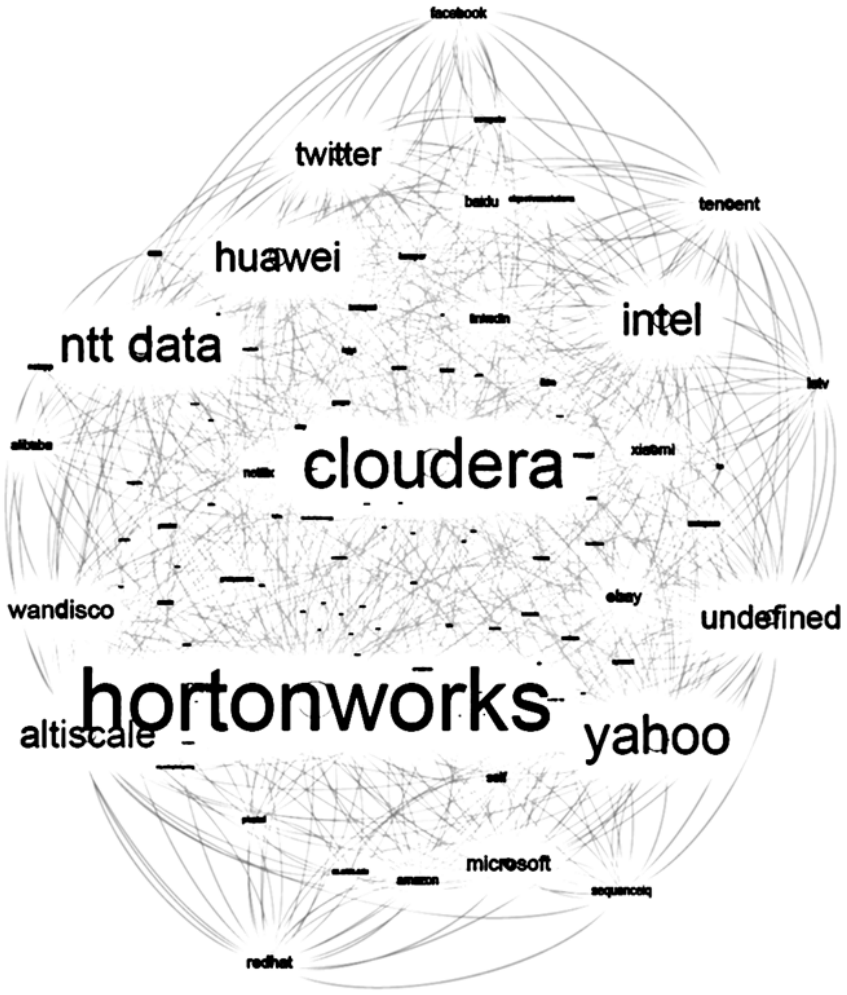


Figure 3: Visualization of the comments-network. Labels are of firms and of relative size of their weighted out-degree to other firms in each network.

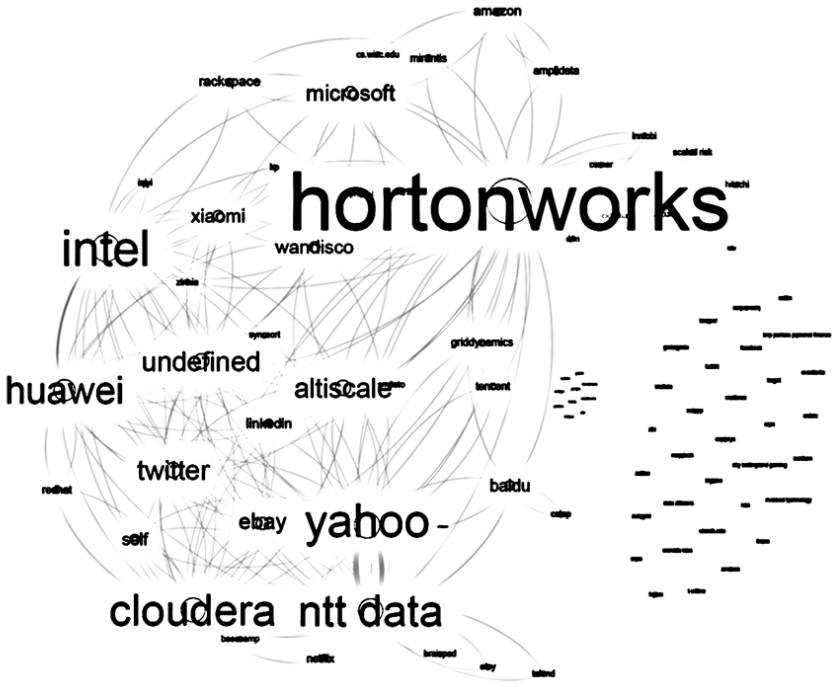


Figure 4: Visualization of the patch-network. Labels are of firms and of relative size of their weighted out-degree to other firms in each network.

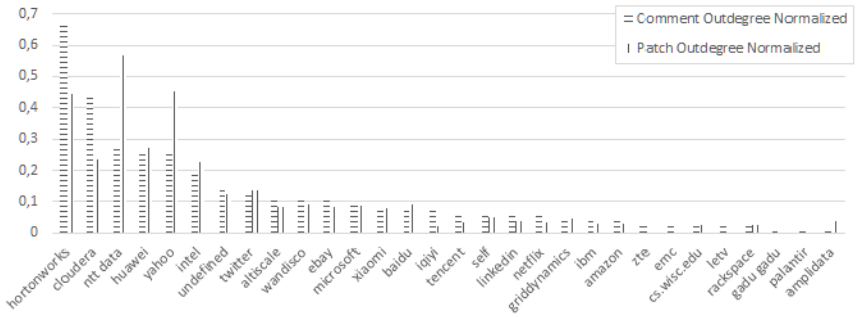


Figure 5: Outdegree centrality (normalized) of the top 30 most influential firms across both patch- and comments-network.

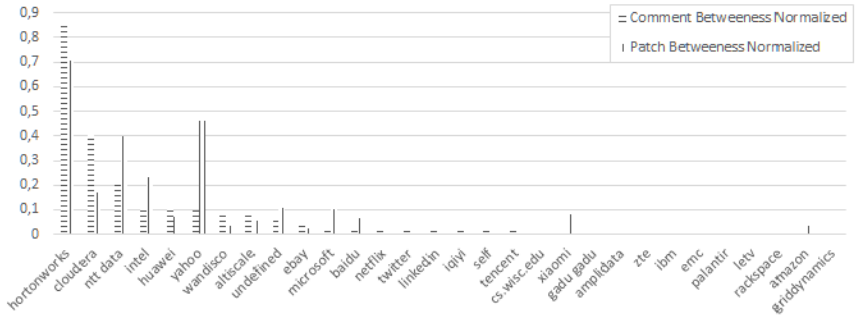


Figure 6: Betweeness centrality (normalized) of the top 30 most influential firms across both patch- and comments-network.

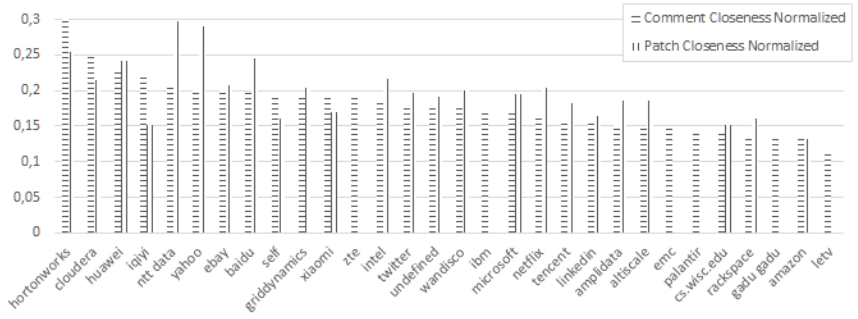


Figure 7: Closeness centrality (normalized) of the top 30 most influential firms across both patch- and comments-network.

stakeholders have a notable difference in centrality between the two networks, such as NTT Data and Yahoo for example. Some stakeholders (e.g., ZTE, IBM and EMC) are completely disconnected in the patch-network while the opposite is true for the comments-network.

Comparing the three centrality measures, both similarities and differences may be found. Although it has a higher activity in the comments-network, Hortonworks has high influence in regards to both technical and social interaction, if all three centrality measures are taken into account. This indicates that Hortonworks has a high impact in regards to what is implemented and how. This firm can be classified as well-connected both directly and indirectly, and has a good position to act as an authority in regards to information spread and coordination. NTT Data and Yahoo both clearly have a higher degree of activity and influence in the patch-network. As with Hortonworks, they also have a similar distribution among all the three measures. This may indicate that they have a high impact in regards to what is implemented and how, but focus their resources on contributing technical implementation suggestions and solutions. As with Hortonworks, they can be classified as well connected both directly and indirectly, and have a good position to act as an authority in regards to information spread and coordination. Regarding the out-degree centrality, a group of stakeholders forms just below the top. This group includes Twitter, Altiscale, Wandisco, EBay, Microsoft, Xiamoi and Baidu, and can be recognized by their medium level of influence and an even focus on technical and social interaction. In contrast to the out-degree centrality, their betweeness centrality differs. For example, Twitter has a relatively low betweeness centrality for both the comments- and patch-networks. This result might imply that have a low number of direct collaborations with other stakeholders - despite the fact that they are influential and active in contributing technically and socially. Twitter therefore has a less central position in regards to coordination, and accumulation and brokerage of information.

A Particular Stakeholder's Influence and Area of Interest

In this example we investigate the agenda of Wandisco (**S1**), which in earlier work was identified as the only infrastructure provider among the top ten most influential stakeholders in the patch-network [108]. In the overview presented in previous example, this was confirmed also in regards to the comments-network (see Fig. 5- 7). Wandisco entered the Apache Hadoop ecosystem in 2012 by acquiring AltoStar. Their product is a platform that allows for distribution of data over multiple Apache Hadoop clusters. They have 14 active developers in the investigated set of releases in regards to comments and patch-contributions. One developer is also a member of the PMC and Committers group. To get a hint about their interests, we investigate if they have shown a special focus in regards to any of the four modules of Apache Hadoop: Common, HDFS, YARN, and MapReduce (**S2**). We focus our attention on requirements included in releases R2.2-R2.7. In regards

S3-4, they are identical to previous example. When creating the interaction networks (**S5**), we generate one patch- and one comments-network for each of the four modules.

In the influence analysis (**S6**), we look at out-degree to get a view of their activity and comprehension of their relative influence in regards to the modules. We present the values for binary out-degree in table 5 and for weighted out-degree in table 6. The former specifically indicates the number of other stakeholders that Wandisco has interacted with, and the latter a better relative measure of their influence. As can be noticed for values regarding the patch-network we can conclude that Wandisco has a specific interest in the HDFS module, which narrows down the analysis made in earlier work [108]. The out-degree values for the comments network further confirms a specific interest in the HDFS module with a relative ranking of 5 and 6 respectively out of 48. Some interest can also be observed for the Common module.

Table 5: Binary out-degree for Wandisco for each of the four modules in the Apache Hadoop projet: Common, HDFS, YARN and MAPREDUCE. Values are aggregated for releases R2.2-2.7 and per network type. Relative ranking is presented within parenthesis.

	Common	HDFS	YARN	Mapreduce
Comments	11 (11/64)	20 (5/48)	4 (32/59)	1 (33/39)
Patches	0	5 (7/24)	0	0

Table 6: Weighted out-degree for Wandisco for each of the four modules in the Apache Hadoop projet: Common, HDFS, YARN and MAPREDUCE. Values are aggregated for releases R2.2-2.7 and per network type. Relative ranking is presented within parenthesis.

	Common	HDFS	YARN	Mapreduce
Comments	1.87 (12/64)	2,82 (6/48)	0.73 (19/59)	0.24 (26/39)
Patches	0	2.97 (7/24)	0	0

Regarding collaboration, we limit our analysis to the HDFS module as this is where their main interest lies. In regards to the patch-network there are only five collaborators, as indicated by the binary out-degree in table 5. These consist of Hortonworks, Huawei, Intel, Yahoo, and Intel. In regards to comments-network, Wandisco had interacted with 20 other stakeholders. Out of these the Hortonworks, Cloudera, Intel, Pivotal and Yahoo were top five in regards to number of comments made by Wandisco on common issues, see 7. The table further presents the weight of the outgoing edge from Wandisco to each respective stakeholder. In the example

of Pivotal, this may be interpreted as Wandisco having made 53 percent of the total number of comments on issues where both Wandisco and Pivotal has collaborated on.

Table 7: Top collaborators with Wandisco in regards to number of comments by Wandisco on issues related to the HDFS module.

Stakeholder	Number of comments	Total number of comments	Weight
Hortonworks	227	1109	0.20
Cloudera	98	663	0.15
Intel	91	679	0.13
Pivotal	42	79	0.53
Yahoo	34	313	0.11

4.3 Analytical Validation of Stakeholder

To analytically validate [73] the framework and its concepts of influence, as well as the proposed network centrality measures that quantify stakeholders' influence on the RE process in an OSS ecosystem, we investigate the strength of correlations between these measures (see table 2) and two measures that are to represent a positive performance for a stakeholder. I.e., that the assumed influence they have had has been used to a positive end in regards to their agenda on the ecosystem's RE process. We define these measures as follows:

P1 Refers to the proportion of issues a stakeholder has reported and that has been included in a release, relative to all included issues. It is intended to describe a stakeholder's ability to get its own requirements prioritized and released.

P2 Refers to the proportion of issues that a stakeholder has interacted with (commented on or submitted a patch to respectively), and that has been included in a release, relative to all included issues. It is intended to describe a stakeholder's ability on get requirements prioritized and released that they have a stake in, but not necessarily having reported themselves. This stake is represented by the action that the stakeholder has shown towards an issue in form of an interaction (i.e., a comment or patch contribution).

In table 8 and 9 we present the results from a pair-wise correlation analysis between the centrality measures and the two performance measures for the comments and patch-networks respectively. The analysis covers 3997 issues that were included in releases R2.2-R2.7 of the Apache Hadoop project. Performance measures P1 and P2 are covered by column 1 and 2 respectively. All except the first are centrality measures and described more extensively in table 2. The number

of authors of a stakeholder regards unique authors of comments and patches respectively to issues that has been included in one of the releases R2.2-R2.7. The measure is included to add a complementary view to the network-based centrality measures as to how much activity a stakeholder has in the ecosystem.

As distributions are skewed and non-linear, we need to consider a rank-based correlation measure. Further, due to that there are edges present (i.e., two stakeholders having equal values in regards to an activity or performance measure), Kendall's tau is chosen over Spearman's tau. Kendall's tau is a non-parametric correlation measure and robust as it considers strength and direction of the monotonic relationship between two variables. To interpret the value of tau we consider Hopskins' scale [75]: trivial (0-0.1), minor (0.1-0.3), moderate (0.3-0.5), large (0.5-0.7), very large (0.7-0.9), and almost perfect (0.9-1).

Table 8 shows Kendall's tau correlations for the patch-network. For both performance measure, all centrality measures show a large correlation with tau > 0.6 except for betweeness centrality (n = 93 and 110 respectively). When comparing binary and weighted measures both show similar values. It is the static measure number of authors that shows the strongest correlation with tau = 0.678 in regards to the relative proportion of reported issues that are included. Same measure shows a very large correlation with tau = 0.877 in regards to the relative proportion of issues that a stakeholder has submitted a patch to and that has been included.

Table 9 shows Kendall's tau correlations for the comments-network. For the first performance measure (n = 93), the binary betweeness centrality shows a very large correlation with tau = 0.743. All other measures show a large correlation with tau > 0.65 except the weighted closeness centrality. For the second performance measure (n = 143), all centrality measures show a very large correlation with tau > 0.7 except the betweeness centrality and the weighted closeness centrality. When comparing binary and weighted centrality measures, the former generally has a higher correlation. As seen in the patch-network, the static measure number of authors the highest correlation with tau = 0.870.

Table 8: Pair-wise correlation analysis between activity and performance measures for the comments-network. Kendall's tau is used with n(P1) = 93 and n(P2) = 143. All correlations have $p < 0.001$.

	Reported Issues Included (P1)	Issues Interacted With (P2)	Mean	Min	Max
Reported Issues Included	1,000	0,456	23,590	0,000	1203,000
Issues Interacted With	0,757	1,000	45,100	0,000	2264,000
Number of Authors	0,672	0,869	3,227	1,000	51,000
Out-degree (binary)	0,692	0,745	6,811	0,000	95,000
Out-degree (weighted)	0,659	0,727	0,954	0,000	23,064
In-degree (binary)	0,692	0,745	6,811	0,000	95,000
In-degree (weighted)	0,654	0,732	0,954	0,000	11,631
Betweeness (binary)	0,743	0,673	112,600	0,000	6263,100
Betweeness (weighted)	0,675	0,594	140,900	0,000	8416,000
Closeness (binary)	0,693	0,746	0,003	0,000	0,007
Closeness (weighted)	0,455	0,571	0,004	0,000	0,011
Eigenvector (weighted)	0,685	0,732	0,163	0,000	1,000

Table 9: Pair-wise correlation analysis between activity and performance measures for the patch-network. Kendall's tau is used with n(P1) = 93 and n(P2) = 110. All correlations have $p < 0.001$.

	Reported Issues Included (P1)	Issues Interacted With (P2)	Mean	Min	Max
Reported Issues Included	1,000	0,572	30,670	0,000	1203,000
Issues Interacted With	0,781	1,000	32,240	0,000	1218,000
Number of Authors	0,677	0,877	3,227	1,000	51,000
Out-degree (binary)	0,666	0,634	1,645	0,000	28,000
Out-degree (weighted)	0,664	0,636	0,964	0,000	18,280
In-degree (binary)	0,664	0,633	1,664	0,000	28,000
In-degree (weighted)	0,641	0,613	0,982	0,000	14,143
Betweeness (binary)	0,571	0,520	19,860	0,000	965,410
Betweeness (weighted)	0,586	0,535	27,820	0,000	822,000
Closeness (binary)	0,654	0,631	0,004	0,000	0,020
Closeness (weighted)	0,643	0,615	0,004	0,000	0,021
Eigenvector (weighted)	0,669	0,635	0,103	0,000	1,000

Although not generally strong, the identified correlations do indicate an association between the activity and performance measures which aligns with what literature states on about the centrality measures and how they may be an indication of influence.

5 Discussion

In a closed setting, the RE process is centered on the firm which elicits the requirements from its stakeholders, whether it being bespoke or market-driven RE [142]. Conceptually, this setting could be resembled to a wheel with the firm constituting the center, and the spokes representing its stakeholders. This is a classical interpretation in stakeholder theory where the firm can consider each stakeholder separately in a dyadic relationship [53]. Even though the stakeholders are the ones from which the requirements are elicited, and depending on importance can affect how they are prioritized, it is still the firm that in the end has the final say in regards to the RE process. In an open setting, as the case of an OSS ecosystem, the focal firm have to participate in an extended RE process. In this case, the firm is no longer the vantage point. Instead, it is now part of a set of stakeholders who collaborate in regards to e.g., requirements elicitation and prioritization. Hence, they now have to start to consider what level of control they need on the ecosystem's RE process and adapt accordingly gain and maintain the needed position in the ecosystem's governance structure [5].

This highlights a distinct difference between the closed and open settings; in the former the firm could focus on the RE process while in the latter they need to also consider strategic aspects in order to control the RE process. The firm no longer has a final say, as even those with a benevolent dictator or project leader position [128] has to consider the will of the ecosystem to avoid the ultimate threat a fork [163]. Control is much more delicate to both gain and practice than in

a corporate environment. It is gained through creating a symbiotic relationship with the ecosystem and practiced through influence [35]. As in Apache Hadoop, meritocracy is a common culture in many OSS ecosystems, meaning that a firm wishing to increase its influence and rise in the governance structure need to be active and contribute back.

Once a firm involved in an OSS ecosystem has realized the potential need of control on the ecosystem's RE process, and the need to build up its influence on it, there comes the question of how. Unstructured engagement may cause differential intellectual property to be given away, but also risk spending resources where there is a limited return of investment in terms of influence and internal business incentives. Firms therefore need to start thinking of what they should contribute and when, also referred to as contribution strategies [175]. These strategies make up a pivotal part of a firm's requirements scoping and management processes which constitutes the main links between a firm's internal RE process and that in of the OSS ecosystem [175]. To answer questions such as what to contribute and where to spend resources, firms need to align their requirements scoping and management processes with their internal product road-map [92] and business requirements [173].

On the opposite side of the coin to the focal firm's agenda, are the different agendas of the OSS ecosystem. Some may be aligned, some not. Further, the stakeholders owning these agendas may have different levels of influence in the OSS ecosystem which may affect whose agenda is valued most, hence what requirements gets prioritized and realized. This highlights the strategic importance of stakeholder identification and analysis, both as an input a firm's contribution strategies in terms of building influence, but also how to use the influence in the ecosystem. The three questions stipulated by Frooman [55] frames this further; firms need to identify and characterize present stakeholders in terms of their influence, identify their agendas primarily in terms of alignment with one's own, and how they are planning to achieve it. The latter of the three is important as it tells the focal firm how they should respond in terms of contribution strategies and interaction.

The framework described here enables the characterization of stakeholders' collaboration and influence within the OSS ecoystem. These attributes can be leveraged as power and control of the RE process in the OSS ecosystem to maximize return on investment in relation to the firm's internal product roadmap [92] and business requirements [173]. The purpose however is not to provide a users' manual for the stakeholder analysis process. The framework is intended to help firms systematically structure this process and enable them to answer questions such as those stipulated by Frooman [55]. Whatever the question of interested or goal, it should be defined from start and used to guide the analysis process in the creation of the stakeholder interaction networks and the following influence analysis. Examples could include the identification of potential partnerships in regards to a certain set of functionality, identification of potential competitors to provide

input on whether a software artifact is differential or commodity, affecting if it should be contributed or not. A further example could be to learn how to increase one's influence by identifying and analyzing a key stakeholder in terms of who they collaborate with and how they invest resources.

6 Threats to Validity

We refer to the four aspects of validity in regards to a case study as proposed by Runeson et al. [146]: *construct*, *internal* and *external validity*, and *reliability*.

Construct validity. Refers to what extent the researchers study what they had set out according to their intentions and research questions. One possible threat is that the centrality measures used may not be interpreted in terms of influence. We address this threat by motivating choice of measures based on literature and empirical work, as well as analytically with correlation analysis against two performance measures that are defined based on a positive performance in regards to a firm's impact on a the RE process in an OSS ecosystem. Defining a positive performance in may be considered troublesome as it may be very contextual for each firm. However, we believe that inclusion of one's issues may be considered as a reasonable indicator of success.

A related threat is that we consider issues in general as "requirements", which may be further extended in our reasoning of requirements artifacts in general. This is based on the nature of RE in OSS as informal and decentralized [45]. Requirements consists of fragmented representations, such as issues, mail-thread discussions and commits [149]. Further action against this threat could include textual and natural language processing of the content in each of the requirements artifacts. This is a vibrant topic in the research field of mining software repositories. However, we consider this topic as out of scope of our framework as we focus on the stakeholder analysis process in its form and structure. We do acknowledge the topic as complementary quality aspects that should be further researched and integrated with our proposed framework in future research.

A further threat concerns the determination of organizational affiliation of individuals in the OSS ecosystem [20]. We adopted a heuristic approach as suggested by earlier research [13, 62], starting with an analysis of email sub-domains and complementing with second and third level sources such as social network sites as LinkedIn and Facebook, as well as blogs, community communication (e.g., comment-history, mailing-lists, IRC logs), web articles and firm websites. We acknowledge this is a delicate and complex process that is best mitigated by "knowing" the ecosystem and actively interacting with its communication channels. In the framework we recommend using a mix-method triangulation with both qualitative and quantitative approaches.

Internal validity. Refers to the risk of unknown confounding factors affecting the results. One threat in this regards the assumed association between a firm's

influence on the RE process of an OSS ecosystem, and the performance measures considering the inclusion of issues. This is a concern since it might very well be that a firm's influence may not be the main cause of an issue being implemented and released. To suggest a causality relationship would require a more thorough analysis and data processing is needed. The main reason for why we have chosen the two performance measures is to validate our proposed approach to quantify influence. We acknowledge that further performance measures can be used for this characterization.

External validity. Refers to what extent the findings of the research are generalizable outside the specific case. The framework is contextualized to consider the informal and decentralized characteristics of RE in OSS framework, as is the underlying foundation in regards to social network analysis methodology and constructs. However, OSS ecosystems may of course differ in regards to informality as pointed out by Ernst and Murphy [45]. Hence, in future work the framework should be applied on further OSS ecosystems, and on firms that are situated in the problem context. This falls natural in the design science approach as it is an iterative search process for an artifact that will solve the stated problem [73].

Reliability. Refers to what extent the generation of the research and its findings are dependent on the original researcher. Sources and inspiration to the framework is described as is the process in how the framework and built up. Due to the data originating from an OSS ecosystem, it is generally available to anyone. Analysis details in regards to creation of networks and application of centrality measures are thoroughly described. Scripts used to generate the networks, their measures and to perform the presented correlations are all available as well for public use and scrutiny.

7 Conclusions

In this study we present a framework to help firms involved in OSS ecosystems to systematically structure their stakeholder analysis process of the ecosystems. The framework focus on characterizing stakeholders according to their level of influence on the ecosystem's RE process. This is an important attribute due to the collaborative and informal nature of the OSS ecosystem's RE processes. The motive behind the framework is to enable firms to identify conflicting and aligning agendas, and how to respond accordingly by adapting its contribution strategies to build and use it its influence in a way that align with internal product road-maps and business requirements.

In future work, we aim to refine and validate the framework qualitatively through further design cycles involving additional OSS ecosystems but adding the expert opinions from involved firms. Further, we aim to investigate how the stakeholder analysis processes resulting from this framework may be used as an input to the construction and execution of firms' contribution strategies [175].

BIBLIOGRAPHY

BIBLIOGRAPHY

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

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

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

[4] Aybüke Aurum and Claes Wohlin. *A Value-Based Approach in Requirements Engineering: Explaining Some of the Fundamental Concepts*, pages 109–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[5] Alfred Baars and Slinger Jansen. *A Framework for Software Ecosystem Governance*, pages 168–180. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

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

[7] George A. Barnett. *Encyclopedia of social networks*. Sage Publications, 2011.

[8] Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, 2004.

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

[10] Tanmay Bhowmik, Nan Niu, Prachi Singhania, and Wentao Wang. On the role of structural holes in requirements identification: An exploratory study on open-source software development. *ACM Transactions of Management Information Systems*, 6(3):10:1–10:30, September 2015.

[11] Mattia Bianchi, Alberto Cavaliere, Davide Chiaroni, Federico Frattini, and Vittorio Chiesa. Organisational modes for open innovation in the bio-pharmaceutical industry: An exploratory analysis. *Technovation*, 31(1):22–33, 2011.

[12] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. Mining email social networks. In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, MSR '06, pages 137–143, New York, NY, USA, 2006. ACM.

[13] 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, Piscataway, NJ, USA, 2012. IEEE Press.

[14] Elizabeth Bjarnason, Michael Unterkalmsteiner, Emelie Engström, and Markus Borg. *An Industrial Case Study on Test Cases as Requirements*, pages 27–39. Springer International Publishing, Cham, 2015.

[15] Elizabeth Bjarnason, Krzysztof Wnuk, and Björn Regnell. Requirements are slipping through the gaps: A case study on causes and effects of communication gaps in large-scale software development. In *2011 IEEE 19th International Requirements Engineering Conference*, pages 37–46, Aug 2011.

[16] Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, pages 1170–1182, 1987.

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

[18] Ulrik Brandes. A faster algorithm for betweenness centrality*. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[19] Daniel J. Brass. Being in the right place: A structural analysis of individual influence in an organization. *Administrative science quarterly*, pages 518–539, 1984.

[20] Jakob Buis and Slinger Jansen. Identifying companies in open source software ecosystems. *Technical Report Series*, UU-CS-2015-011, 2015.

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

[22] Marjolein C.J. Caniels and Cees J. Gelderman. Purchasing strategies in the kraljic matrix: A power and dependence perspective. *Journal of Purchasing and Supply Management*, 11(2):141 – 155, 2005.

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

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

[25] Henry Chesbrough. *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press, 2006.

[26] Henry Chesbrough and Melissa M. Appleyard. Open innovation and strategy. *California management review*, 50(1):57–76, 2007.

[27] Henry Chesbrough and Adrienne Kardon Crowther. Beyond high tech: early adopters of open innovation in other industries. *R&d Management*, 36(3):229–236, 2006.

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

[29] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board. *IEEE Recommended Practice for Software Requirements Specifications*. Institute of Electrical and Electronics Engineers, 1998.

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

[31] Kevin Crowston and James Howison. The social structure of free and open source software development. *First Monday*, 10(2), 2005.

[32] Daniela S. Cruzes, Tore Dybå, Per Runeson, and Martin Höst. Case studies synthesis: Brief experience and challenges for the future. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 343–346, Sept 2011.

[33] 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*, 20(6):1634–1665, 2015.

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

[35] 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.

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

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

[38] Daniela Damian, Sabrina Marczak, and Irwin Kwan. Collaboration patterns and the impact of distance on awareness in requirements-centred social networks. In *15th IEEE International Requirements Engineering Conference*, pages 59–68. IEEE, 2007.

[39] Sherae Daniel, Likoebe Maruping, Marcelo Cataldo, and James Herbsleb. When cultures clash: Participation in open source communities and its implications for organizational commitment. 2011.

[40] 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.

[41] Anh Nguyen Duc, Daniela S. Cruzes, Claudia Ayala, and Reidar Conradi. Impact of stakeholder type and collaboration on issue resolution time in oss projects. In *Open Source Systems: Grounding Research*, pages 1–16. Springer, 2011.

[42] 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.

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

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

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

[46] Katherine Faust. Centrality in affiliation networks. *Social Networks*, 19(2):157 – 191, 1997.

[47] John Favaro. Managing requirements for business value. *IEEE Software*, 19(2):15–17, Mar 2002.

[48] Daniel Mendez Fernandez, 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.

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

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

[51] Bent Flyvbjerg. Five misunderstandings about case-study research. In *Qualitative Research Practice*, pages 390–404. SAGE, concise paperback edition, 2007.

[52] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[53] Robert E. Freeman. *Strategic management: A stakeholder approach*. Cambridge University Press, 1984.

[54] Samuel Fricker. *Requirements Value Chains: Stakeholder Management and Requirements Engineering in Software Ecosystems*, pages 60–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[55] Jeff Frooman. Stakeholder influence strategies. *Academy of management review*, 24(2):191–205, 1999.

[56] G.R. Gangadharan, Lorna Uden, and Paul Oude Luttighuis. Sourcing Requirements and Designs for Software as a Service. *International Journal of Systems and Service-Oriented Engineering*, 6(1):1–16, jan 2016.

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

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

[59] Eva Geisberger, Manfred Broy, Brian Berenbach, Juergen Kazmeier, Daniel Paulish, and Arnold Rudorfer. Requirements engineering reference model (rem). *Technische Universität München, München*, 2006.

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

[61] Martin Glinz and Roel J. Wieringa. Guest editors' introduction: Stakeholders in requirements engineering. *IEEE Software*, 24(2):18–20, 2007.

[62] 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.

[63] Abbie Griffin. Metrics for measuring product development cycle time. *Journal of product innovation management*, 10(2):112–125, 1993.

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

[65] Robert A. Hanneman and Mark Riddle. Introduction to social network methods, 2005.

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

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

[68] Lile P. 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, September 2008.

[69] Frank Hecker. Setting up shop: The business of open-source software. *IEEE software*, 16(1):45, 1999.

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

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

[72] 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.

[73] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, March 2004.

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

[75] Will G. Hopskins. A scale of magnitudes for the effect statistics: A new view of statistics, August 2006.

[76] Google Project Hosting. Gerrit code review source code repository. `https://code.google.com/p/gerrit/wiki/Source?tm=4`. Accessed: 2014-06-24.

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

[78] Stefan Husig 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.

[79] Serhan Ili, Albert Albers, and Sebastian Miller. Open innovation in the automotive industry. *R&d Management*, 40(3):246–255, 2010.

[80] 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.

[81] 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.

[82] 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, May 2009.

[83] Chris Jensen and Walt Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *29th International Conference on Software Engineering*, pages 364–374. IEEE, 2007.

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

[85] Lena Karlsson, Åsa G. Dahlstedt, Johan Natt och Dag, Björn Regnell, and Anne Persson. Challenges in market-driven requirements engineering-an industrial interview study. In *8th International Workshop on Requirements Engineering: Foundation for Software Quality*, 2002.

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

[87] Hans-Bernd Kittlaus and Peter N. Clough. *Software Product Management and Pricing: Key Success Factors for Software Organizations*. Springer Berlin Heidelberg, 2008.

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

[89] Dan Knight, Robert M. Randall, Amy Muller, Liisa Välikangas, and Paul Merlyn. Metrics for innovation: guidelines for developing a customized suite of innovation metrics. *Strategy & Leadership*, 33(1):37–45, 2005.

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

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

[92] Marko Komssi, Marjo Kauppinen, Harri Töhönen, Laura Lehtola, and Alan M. Davis. Roadmapping problems in practice: value creation from the perspective of the customers. *Requirements Engineering*, 20(1):45–69, 2015.

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

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

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

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

[97] Paula Laurent and Jane Cleland-Huang. *Lessons Learned from Open Source Projects for Facilitating Online Requirements Processes*, pages 240–255. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[98] 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.

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

[100] 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.

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

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

[103] 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.

[104] Soo Ling Lim, Daniele Quercia, and Anthony Finkelstein. Stakenet: using social networks to analyse the stakeholders of large-scale software projects. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pages 295–304. ACM, 2010.

[105] Johan Linåker, Maria Krantz, and Martin Höst. *On Infrastructure for Facilitation of Inner Source in Small Development Teams*, pages 149–163. Springer International Publishing, Cham, 2014.

[106] Johan Linåker, Husan Munir, Per Runeson, Björn Regnell, and Claes Schrewelius. *A Survey on the Perception of Innovation in a Large Product-Focused Software Organization*, pages 66–80. Springer International Publishing, Cham, 2015.

[107] Johan Linåker, Björn Regnell, and Hussan Munir. Requirements engineering in open innovation: a research agenda. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 208–212. ACM, 2015.

[108] Johan Linåker, Patrick Rempel, Björn Regnell, and Patrick Mäder. *How Firms Adapt and Interact in Open Source Ecosystems: Analyzing Stakeholder Influence and Collaboration Patterns*, pages 63–81. Springer International Publishing, Cham, 2016.

[109] Johan Linåker, Sardar Saluman, Rafael Maiani de Mello, and Martin Höst. Guidelines for conducting surveys in software engineering, 2005.

[110] Johan Linåker and Krzysztof Wnuk. Requirements analysis and management for benefiting openness (rambo). In *Proceedings of the 2016 International Workshop on Software Product Management*. IEEE, 2016.

[111] 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.

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

[113] Andrey Maglyas and Samuel A. Fricker. *The Preliminary Results from the Software Product Management State-of-Practice Survey*, pages 295–300. Springer International Publishing, Cham, 2014.

[114] Konstantinos Manikas and Klaus Marius Hansen. Software ecosystems–a systematic literature review. *Journal of Systems and Software*, 86(5):1294–1306, 2013.

[115] Sabrina Marczak, Daniela Damian, Ulrike Stege, and Adrian Schröter. Information brokers in requirement-dependency social networks. In *2008 16th IEEE International Requirements Engineering Conference*, pages 53–62, Sept 2008.

[116] Juan Martinez-Romo, Gregorio Robles, Jesus M. Gonzalez-Barahona, and Miguel Ortuño-Perez. Using social network analysis techniques to study collaboration between a floss community and a company. In *Open Source Development, Communities and Quality*, pages 171–186. Springer, 2008.

[117] MetricsGrimoire. CVSAnalY. `http://metricsgrimoire.github.io/CVSAnalY/`. Accessed: 2014-07-17.

[118] Ronald K. Mitchell, Bradley R. Agle, and Donna J. Wood. Toward a theory of stakeholder identification and salience: Defining the principle of who and what really counts. *Academy of management review*, 22(4):853–886, 1997.

[119] 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.

[120] Charlotte Möller and Madeleine Wahlqvist. Critical success factors for innovative performance of individuals: A case study of scania. *Management*, 39(5):1155–1161.

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

[122] 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.

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

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

[125] Hussan Munir, Johan Linåker, Krzysztof Wnuk, Per Runeson, and Björn Regnell. Open innovation using open source tools: A case study at sony mobile. *Submitted to Software Engineering Journal*.

[126] Hussan Munir and Per Runeson. Software testing in open innovation: An exploratory case study of the acceptance test harness for jenkins. In *Proceedings of the 2015 International Conference on Software and System Process*, ICSSP 2015, pages 187–191, New York, NY, USA, 2015. ACM.

[127] Hussan Munir, Krzysztof Wnuk, and Per Runeson. Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering*, pages 1–40, 2015.

[128] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. Evolution patterns of open-source software systems and communities. In *Proceedings of the international workshop on Principles of software evolution*, pages 76–85. ACM, 2002.

[129] Barry J. Nalebuff and Adam M. Brandenburger. Co-opetition: Competitive and cooperative business strategies for the digital economy. *Strategy & leadership*, 25(6):28–33, 1997.

[130] Mark Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical review E*, 64(1):016132, 2001.

[131] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

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

[133] Ohloh.net. The jenkins gerrit trigger plugin open source project. `https://www.ohloh.net/p/gerrit-trigger-plugin`. Accessed: 2014-07-08.

[134] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, 2010.

[135] Alma Orucevic-Alagic and Martin Höst. Network analysis of a large scale open source project. In *40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 25–29, Verona, Italy, 2014. IEEE.

[136] Carla Pacheco and Ivan Garcia. A systematic literature review of stakeholder identification methods in requirements elicitation. *Journal of Systems and Software*, 85(9):2171–2181, 2012.

[137] 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.

[138] 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.

[139] Karl Michael Popp. Leveraging open source licenses and open source communities in hybrid commercial open source business models. In *IWSECO@ ICSOB*, pages 33–40, 2012.

[140] Coimbatore K. Prahalad and Mayuram S. Krishnan. The new meaning of quality in the information age. *Harvard business review*, 77(5):109–18, 1998.

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

[142] Björn Regnell and Sjaak Brinkkemper. Market-driven requirements engineering for software products. In *Engineering and managing software requirements*, pages 287–308. Springer, 2005.

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

[144] 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.

[145] Timothy J. Rowley. Moving beyond dyadic ties: A network theory of stakeholder influences. *Academy of management Review*, 22(4):887–910, 1997.

[146] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.

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

[148] Soumodip Sarkar and Ana Costa. Dynamics of open innovation in the food industry. *Trends in Food Science & Technology*, 19(11):574–580, 2008.

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

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

[151] Stijn Schuermans, Andreas Constantinou, and Michael Vakulenko. Assymetric business models: The secret weapon of software-driven companies, 2014.

[152] 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.

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

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

[155] 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.

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

[157] Jose Teixeira and Tingting Lin. Collaboration in the open-source arena: the webkit case. In *Proceedings of the 52nd ACM conference on Computers and people research*, pages 121–129. ACM, 2014.

[158] Jose Teixeira, Gregorio Robles, and Jesús M González-Barahona. Lessons learned from applying social network analysis on an industrial free/libre/open source software ecosystem. *Journal of Internet Services and Applications*, 6(1):1–27, 2015.

[159] Richard Torkar, Pau Minoves, and Janina Garrigós. Adopting free/libre/open source software practices, techniques and methods for industrial use. *Journal of the Association for Information Systems*, 12(1):88–122, 2011.

[160] 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.

[161] Inge Van De Weerd, Sjaak Brinkkemper, Richard Nieuwenhuis, Johan Versendaal, and Lex Bijlsma. Towards a reference framework for software product management. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 319–322. IEEE, 2006.

[162] 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.

[163] 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.

[164] 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.

[165] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[166] Florian Weikert and Dirk Riehle. A model of commercial open source software product features. In *International Conference of Software Business*, pages 90–101. Springer, 2013.

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

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

[169] Joel West and Marcel Bogers. Leveraging external sources of innovation: A review of research on open innovation. *Journal of Product Innovation Management*, pages n/a–n/a, 2013.

[170] 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.

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

[172] 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.

[173] Karl Wiegers and Joy Beatty. *Software requirements*. Pearson Education, 2013.

[174] Roel J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[175] Krzysztof Wnuk, Dietmar Pfahl, David Callele, and Even-André Karlsson. How can open source software development help requirements management gain the potential of open innovation: an exploratory study. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 271–280. ACM, 2012.

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