Guiding Development of Contribution and Community Strategies in Open Source Software Requirements Engineering

Johan Linåker

Doctoral Dissertation, 2019
Department of Computer Science
Lund University

LU-CS-DISS 2019-03
Doctoral Dissertation 63, 2019

ISBN: 978-91-7895-173-4 (Printed)
ISBN: 978-91-7895-174-1 (Electronic)
ISSN: 1404-1219

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, 2019

ABSTRACT

Background: For software-intensive organizations, Open Source Software (OSS) may provide a pivotal building block in business models and strategies, product and service offerings, as well as in tool and infrastructure setups. The Requirements Engineering (RE) and development processes of OSS take place inside communities where the focal organization is a stakeholder among many, including competitors. Therefore, to exploit the potential benefits of OSS, an organization has to consider what it shares as OSS and how it engages with the OSS communities. By being too open, an organization may expose itself to risks such as giving away differentiating functionality. On the contrary, being too closed may cause misalignment between an organization's RE process and that of a community.
Objective: The objective of this thesis is two-fold. Firstly, to create guidance for organizations in making decisions of what to share as OSS in line with the organization's business goals. Secondly, to create guidance for how an organization can identify OSS communities where they need to have an influence on the RE process, and how they can gain it, in order to achieve its internal agenda.
Research Methodology: We used a design science research approach, applying empirical software engineering research methods to investigate the problem context and design and validate solution artifacts that may be used as treatments in the problem context. The relevance of the research has been maintained through a close industry collaboration with several case studies and interview surveys.
Results: To address the two objectives, we introduce the two concepts of contribution and community strategy. Contribution strategies answer the questions if a software artifact (e.g., a feature or project) or parts of it should be released as OSS, when in time, and if it should be contributed to an existing OSS community, or if a new community should be established. Community strategies answer the questions what OSS communities an organization considers as important and need to have an influence on in terms of their RE process, and also how this influence may be gained. The thesis offers problem understanding of how organizations reason in terms of these questions, as well as guidance for how the different types of strategies may be developed. In regards to contribution strategies, results also offer guidance on how to create supporting guidelines, processes, and infrastructure on an organizational level.

Conclusion: The results of this thesis are captured in a number of frameworks, models, and methods. These artifacts contribute to an understanding of the problem context and provide design knowledge and exemplars that may be transferred and implemented by organizations in a real-world problem context. Evaluation of such a technology transfer is a topic for future work.

POPULAR SCIENCE SUMMARY

ÖPPEN KÄLLKOD ÖPPNAR NYA DÖRRAR

JOHAN LINÅKER, DEPARTMENT OF COMPUTER SCIENCE

Organisationer inom både privat och offentlig sektor har mycket att vinna på att börja utveckla och dela med sig av sin mjukvara som öppen källkod. Ökad innovation och produktivitet, och likaså nya affärsmöjligheter är några av fördelarna. Men för att kunna ta del av fördelarna behöver organisationer bestämma vilken mjukvara de delar med sig av, när och hur. Eventuella kostnader och risker behöver övervägas, hanteras och vägas mot det potentiella värdeskapandet. Organisationer kan då skapa tydliga strategier för hur de kan öppna både sig själva och sin mjukvara.

Denna avhandling har undersökt hur organisationer inom privat och offentlig sektor kan utforma dessa typer av strategier. Undersökningarna har resulterat i beslutsstöd bestående av tre delar vilka denna artikel ger en översiktlig introduktion till.

Mjukvara som öppen källkod

När en mjukvara benämns som öppen källkod betyder det att källkoden som mjukvaran är uppbyggd av får användas, läsas och redigeras samt distribueras vidare fritt, givet vissa villkor som finns definierade i en licens. Öppenheten medför att källkoden kan utvecklas transparent tillsammans av både enskilda personer,

företag och offentliga organisationer. Dessa intressenter deltar i utvecklingen av projekten drivna av deras egna motiv och agendor.

Delningsstrategier för öppen källkod

Den första delen av beslutsstödet hjälper berörda organisationer att utveckla vad vi benämner som delningsstrategier. Dessa strategier definierar vilken mjukvara som bör öppnas upp, när och med vem. Med hjälp av beslutsstödet kan organisationer skapa dessa strategier samt utforma nödvändiga besluts- och uppföljningsprocesser som krävs för att implementera dessa.

För att skapa en delningsstrategi bör en organisation dels sätta upp relevanta mål som de önskar uppnå genom att öppna upp sin mjukvara, och dels analysera hur dessa mål ska uppnås och är motiverade i förhållande till identifierade risker, kostnader samt andra komplicerande aspekter. Beslutsstödet pekar ut ett antal mål och aspekter baserat på fallstudier och observationer hos fyra större organisationer som på olika sätt nyttjar och delar med sig utav mjukvara som öppen källkod.

Bland de vanligast förekommande målen som identifierats kan exempelvis nämnas att skapa kostnadsbesparingar genom delat underhåll, öka inflöde av innovativa funktioner, samt förbättra ryktet som arbetsgivare för utvecklare. Ett mer strategiskt mål kan vara att skapa en ny standardlösning för att på så sätt ta sig in på nya marknader och skapa plattformar för tredjepartsprodukter. För privata organisationer kan ytterligare en anledning vara att försvåra affärsmöjligheter för konkurrenter genom att öppna upp mjukvara som är differentierande för bägge parter. Motsatt sätt kan det för offentliga organisationer vara ett sätt att tillgängliggöra en infrastruktur och plattform för att möjliggöra för fler privata organisationer att konkurrera på aktuell marknad.

En av de vanligaste riskerna som identifierats bland privata organisationer är att de skulle råka ge bort sina ”kronjuveler”, det vill säga mjukvara som är differentierande för organisationen och som utgör en viktig konkurrensfördel. Dock är det vanliga fallet att det är en begränsad del av mjukvaran som är så pass värdefull att den bör hållas stängd. Genom att identifiera och separera de delar som ger en konkurrensfördel kan övriga delar fortfarande öppnas upp. För offentliga organisationer kan ambitionen vara den motsatta. Där kan syftet vara att dela den mest differentierande mjukvaran för att skapa störst värde för mjukvarans potentiella användare. En risk är dock att detta potentiellt kan skada vissa organisationers affärsmöjligheter, varför en noggrann analys bör göras innan mjukvaran delas.

En relaterad fråga till vad som bör delas och när, är om mjukvaran ska delas med ett befintligt öppen källkodsprojekt eller om ett nytt bör skapas. Att dela med sig till ett befintligt projekt är ofta att föredra då alternativet är både kostsamt och riskfyllt. För att tryggt kunna dela med sig mjukvara till ett befintligt projekt kan berörd organisation behöva en viss nivå av inflytande över projektets utveckling. Detta då det kan finnas andra intressenter involverade i projektet som har avvikande

agendor och skulle kunna motsätta sig att den aktuella mjukvaran accepteras som ett bidrag till projektet.

Projektstrategier för öppna källkodsprojekt

Den andra delen av beslutsstödet fokuserar därför på att hjälpa organisationer skapa projektstrategier som beskriver hur de bör agera för att få ett inflytande samt identifiera de projekt där inflytandet behövs. Beslutsstödet beskriver hur en projektstrategi ska ta hänsyn både till den affärsmässiga och tekniska kopplingen mellan ett projekt och en organisation, likaså möjligheten att faktiskt bygga upp inflytandet i projektet. Därutöver pekas åtta tillvägagångssätt ut som en organisation sedan kan nyttja för att bygga upp detta inflytande. Inom samtliga tillvägagångssätt är det avgörande att organisationen engagerar sig i utvecklingen av projektet och delar med sig av exempelvis kunskap, mjukvara och andra typer av resurser.

Intressentanalyser av öppna källkodsprojekt

För att underlätta utformande och beslut av delnings- respektive projektstrategier omfattar den tredje delen av beslutsstödet en metod för att utföra intressentanalyser av öppna källkodsprojekt. En sådan analys besvarar frågor kring vilka intressenterna till projektet är, hur stort inflytande de har över projektets utveckling, samt vad deras respektive agenda utgör. Metoden kan därigenom exempelvis hjälpa med att identifiera potentiella samarbetspartners samt konkurrenter, och bygger på att interaktioner mellan individer involverade i utvecklingen av projekt visualiseras och analyseras i form av sociala nätverk. Genom dessa nätverk kan sedan de individer, och därigenom de organisationer, som är de mest centrala eller inflytelserika identifieras och därefter deras respektive agendor undersökas närmre.

Öppen källkod öppnar nya dörrar

Det finns flera faktorer som kan tala både för och emot om en mjukvara bör öppnas upp, tillika när och hur. Dessa bör identifieras, analyseras och vägas mot varandra för att ett välgrundat beslut ska kunna fattas. Vilka faktorer som är relevanta varierar mellan olika sammanhang, exempelvis typen av organisation, dess affärsmodell samt hur aktuell mjukvara och eventuella öppna källkodsprojekt kopplar till denna. Vårt forskningsbaserade beslutsstöd visar hur delnings- och projektstrategier kan skapas och hjälpa berörda organisationer i deras arbete med att öppna upp både sig själva och sin mjukvara, och därmed kunna ta del av de många fördelar som öppen källkod öppnar upp för.

ACKNOWLEDGEMENTS

This work was funded by the Swedish National Science Foundation Framework Grant for Strategic Research in Information and Communication Technology, project Synergies (Synthesis of a Software Engineering Framework for Open Innovation through Empirical Research), grant 621-2012-5354, and the industrial excellence center EASE (Embedded Applications Software Engineering)[1].

Funding aside, I would like to express my deepest gratitude to everyone, without whose help this doctoral 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. A special thanks also goes to Prof. Dr. Daniela Damian at University of Victoria for both her mentorship and support.

Thanks to Dr. Hussan Munir for the long and intellectual discussions ;) And thanks to everyone else, past and present, 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. Also, thanks to Sony Mobile in Lund, as well as the other case organizations 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 wife Christine Cleyton Jørgensen and two wonderful sons Vilhelm and Folke, and also to the rest of my family and friends for making life joyful, great and awesome in all regards.

Cheers!
Johan Linåker

[1] http://ease.cs.lth.se

LIST OF PUBLICATIONS

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

Publications included in the thesis

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

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

III **A Community Strategy Framework – How to obtain Influence on Requirements in Meritocratic Open Source Software Communities?**
Johan Linåker, Björn Regnell and Daniela Damian. Information and Software Technology, 2019, in press.

IV **A Method for Analyzing Stakeholders' Influence on an Open Source Software Ecosystem's Requirements Engineering Process**
Johan Linåker, Björn Regnell and Daniela Damian. Requirements Engineering, 2019, in press.

V **A Contribution Management Framework – What to share as Open Source Software?**
Johan Linåker and Björn Regnell. Unpublished manuscript.

Related Publications

Related publications that are not included in this thesis:

VI **High-level software requirements and iteration changes: a predictive model**
Kelly Blincoe, Ali Dehghan, Abdoul-Djawadou Salaou, Adam Neal, Johan Linåker and Daniela Damian. Empirical Software Engineering, 2018, in press.

VII **Predicting likelihood of requirement implementation within the planned iteration: an empirical study at IBM**
Ali Dehghan, Adam Neal, Kelly Blincoe, Johan Linåker and Daniela Damian. In Proceedings of the 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, pp. 124-134, 2017.

XIII **A Contribution Management Framework for Firms Engaged in Open Source Software Ecosystems-A Research Preview**
Johan Linåker and Björn Regnell. In Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), Essen, Germany, pp. 50-57, 2017.

IX **The open source officer role - experiences**
Carl Eric Mols, Krzysztof Wnuk and Johan Linåker. In Proceedings of the 13th IFIP International Conference on Open Source Systems (OSS), Buenos Aires, Argentina, pp. 55-59, 2017.

X **How firms adapt and interact in open source ecosystems: analyzing stakeholder influence and collaboration patterns**
Johan Linåker, Patrick Rempel, Björn Regnell, Patrick Mäder. In Proceedings of the 22nd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), Gothenburg, Sweden, pp. 63-81, 2016.

XI **Requirements Analysis and Management for Benefiting Openness**
Johan Linåker and Krzysztof Wnuk. In Proceedings of the 9th International Workshop on Software Product Management (IWSPM), Beijing, China, pp. 344-349, 2016.

XII **A survey on the perception of innovation in a large product-focused software organization**
Johan Linåker, Hussan Munir, Per Runeson, Björn Regnell and Claes Schrewelius. In Proceedings of the 6th International Conference of Software Business (ICSOB), Braga, Portugal, pp.66-80, 2015.

XIII **Requirements engineering in open innovation: a research agenda**
Johan Linåker, Björn Regnell, Hussan Munir. In Proceedings of the 1st International Workshop of Open Innovation in Software Engineering (OISE), Estonia, Tallin, pp. 208-212, 2015.

XIV **On infrastructure for facilitation of inner source in small development teams**
Johan Linåker, Maria Krantz, Martin Höst In Proceedings of the 15th International Conference on Product-Focused Software Process Improvement (PROFES), Helsinki, Finland, pp. 149-163, 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
Dr. Hussan Munir was the lead author and together with Johan Linåker 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 II
Johan Linåker was the lead author and together with Dr. 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 III
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.

Paper IV
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.

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

CONTENTS

INTRODUCTION

1 Introduction

Sharing internally developed software as Open Source Software (OSS), as well as partaking in the open development of such software may seem counter-intuitive for some software-intensive organizations [86]. By contributing the software to an existing community, or by releasing it within a new community, the software is commoditized, and free to use, modify, and also distribute given that conditions defined in the software's OSS license are met. Chesbrough explains the rationale with the term Open Innovation (OI) which implies a distributed innovation process where an organization goes beyond its own borders in a process of creating, delivering and capturing value (monetary or non-monetary) [30]. Instead of solely relying on their internal research and development, organizations are encouraged to open up and take advantage of external ideas, resources, and knowledge, e.g., through sourcing or acquisition [37]. In a similar fashion, organizations 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 [37]. Either way, or combined, OI highlights the importance of recognizing the external workforce that resides outside of the organization, and the knowledge and competencies that they possess [222].

In the case of OSS, the external workforce is constituted by the community of actors that through one or more common incentives collaboratively see to the development and maintenance of the OSS [213]. The community may also be framed as a software ecosystem using the definition by Jansen et al. [97] where the OSS constitutes the ”common technological platform” that underpins the relationships and interactions between the actors. Through this analogy, organizations applying OSS from an OI perspective may be seen as members of an OSS ecosystem [141].

From a business model perspective, the OSS may be used as a direct part of an organization's product offering, e.g., through an open core or platform-extension model [217], as a basis for support, subscriptions and professional services [32], or as part of a duel-licensing model [222]. However, it may also be the case that the

value comes indirectly when the OSS is used as an enabler for the organizations' product offerings, e.g., as a development component or as part in the tool and infrastructure setup, supporting the development and delivery of the organization's product [85]. It may also be a combination of such direct and indirect factors. For example, in asymmetric business models, the software is made open in order to capture value from additional products, services and data gathering that is managed through the OSS [195]. The potential value that motivates these different usages of OSS has its foundation in the external workforce that is available in the OSS community [159]. As explained by OI [30], this can help the organization 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 the software and decrease in the time-to-market [205].

To gain the potential benefits, an organization needs to "open up" as hinted by Chesbrough [34]. However, deciding what should be opened-up, or contributed, is a complex matter [86, 159]. Giving away differentiating, or in other ways sensitive intellectual property rights, especially to competitors, may have negative effects on both for existing and future business (e.g., [86, 87, 96, 212, 222, 231]). The timing of a contribution is expressed by Wnuk et al. [231] as a balance between losing a competitive edge and increased maintenance costs. Waiting too long may imply alternative solutions being adopted with the focal organization[2] having the choice to either adapt or maintain their internal solution [157]. Where to contribute is a third challenge. When contributing the software artifact to an existing OSS community, not governed by the organization itself [172], the organization needs to consider the RE process of the community [192]. An option is to release the software as an independent OSS project and build a new community around it, which may require significant investment as well [109, 223]. The answers to *what* should be contributed, *where*, and *when*, are defined in this thesis as a *contribution strategy*.

However, as exemplified, to answer these questions and decide on a contribution strategy a number of complexities may need to be considered. Some of these may not be relevant to all organizations. Contribution processes can, for example, be more liberal in organizations or business units where the OSS used is not considered to provide a competitive advantage [157]. Contextual factors also play a role in terms of how value is perceived for an organization, which motivates why a contribution should be made [128]. By not considering relevant complexities or benefits, an organization may risk making a contribution that could be harmful, or on the opposite, block a contribution that could have been beneficial for the organization and its business goals [231]. Organizations hence need to link their business goals with their contribution strategies, why the first research goal of this thesis is:

[2]The organization which perspective is implied.

RG1: To design a solution that supports software-intensive organizations in developing contribution strategies that align with the organizations' business goals.

If the organization intends to contribute to an existing OSS community, they have to consider participation in the external RE process of the community, and how to bridge such participation with their internal process [131]. In contrast to the latter, the RE process in OSS communities can usually be characterized as being informal and decentralized with a focus on collaboration and transparency [54, 192]. These characteristics further highlight that the focal organization may no longer be the vantage point as the case in their internal RE process [189]. In the OSS community, the focal organization may be considered as a stakeholder among others in the fluctuating and open stakeholder population. This may imply risks of conflicting agendas [159] and difficulty in aligning internal strategies and processes with those of the community [159]. To manage such risks an organization may need to gain and maintain a suitable position in the community's governance structure [10] in order to have the influence needed in regards to the community's RE process [231].

With influence, we refer to the Merriam-Webster dictionary[3] which defines it as *"the power to change or affect someone or something"*. In our context, influence relates to the power of an organization to change or affect a requirement of interest in an OSS community, for example, how a requirement is specified, prioritized, and realized, both short-term in release-planning, and long-term on the road-map [74, 120, 167].

In OSS communities with a meritocratic governance structure [46, 143], either in part or in full [198], influence may be gained by proving merit and earning trust and status within the community [58]. What merit constitutes depends on the context [51, 174], but is generally gained by building an active and symbiotic relationship with the community where an organization dedicates resources, contributes internal requirements and actively participates in the development of the OSS [24, 38, 40, 166, 194, 206]. Besides enabling an organization to contribute, gaining influence in a meritocratic OSS community also offers an opportunity to influence the community's RE process according to the organization's own agenda while competing and collaborating with the other stakeholders in the community [166]. An organization in the problem context may, therefore, have to consider the questions what OSS communities they view as important and need to have an influence on in terms of the RE process, and also how this influence may be gained. Answers to these question are defined in what this thesis refers to as a *community strategy*. As with contribution strategies, organizations should strive to align the community strategies with business goals, why the second research goal of this thesis is:

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

RG2: To design a solution that supports software-intensive organizations in developing community strategies that align with the organizations' business goals.

The challenges infused by an OSS community's stakeholder population illustrates the importance of stakeholder identification and analysis as input to the continuous and complex decision-making process which RE constitutes [8]. Such analysis could help an organization engaged (or thinking of getting engaged) in a community by answering the questions which other stakeholders exist in the community, what are the different stakeholders' agendas, and how do they aim to achieve their respective agendas [65]? Answers may help an organization to identify possible partners and competitors [189], as well as to learn how to adapt its own strategies and processes with the OSS community's and how to build its own influence and position the community's governance structure [10]. The knowledge output can then be leveraged towards other stakeholders through the politics and negotiations that take place in the community's RE process [146]. The third research goal of this thesis is, therefore:

RG3: To design a solution that supports software-intensive organizations in analyzing stakeholders' influence on an Open Source Software community's Requirements Engineering process.

To address *RG1-3*, this thesis and its included papers use a design science research approach which is further explained in Section 2. In Section 3, more background, and related work are presented to provide further contextualization of the research goals as well as the contributions of this thesis. The results of each of the included papers are presented in Section 4, followed by a presentation of the synthesis and main contributions of this thesis in Section 5. Threats to validity and ethical aspects of the research are discussed in Section 6. Lastly, future work is discussed in Section 7, followed by the conclusions and a summary of the main contributions of this thesis in Section 8.

2 Research Approach

This thesis has used a design science research approach within and between *Papers I-V*. Below, a conceptualization of design science as a research approach is first presented to provide a common frame of reference for the rest of the thesis. This is followed by a short description of the case study research method which is used in a majority of the included papers. The research design is then presented of the of the thesis on a general level, and in regards to the individual papers included.

2.1 Conceptualization of Designs Science

Wieringa [228] describes design science as consisting of two parts, the design and investigation of an artifact in a problem context. The artifact is designed to interact with the problem context and provide an improvement. The design part is focused on solving design problems by designing artifacts that may improve the problem contexts for concerned stakeholders. The investigation part is focused on answering knowledge questions about the artifacts in their respective problem contexts. Wieringa views design problems and knowledge questions as two interrelated types of research problems. The former *"calls for a change in the world"*, while the latter calls for knowledge about the world as is [227]. This generates an iteration between the design and investigation, where a design problem renders in knowledge questions or vice versa [228].

Aligning with van Aken [3], the focus of design science can, therefore, be said to investigate and understand a real-world problem in its context and develop knowledge that can be used to design artifacts that may provide an improvement, or solve the problem in question in the real world. Producing such prescriptive knowledge, also referred to as design knowledge [3], aligns with the aim of most software engineering research which is to provide advice for practitioners within the problem context on how to act given various situations [204]. The prescription-driven design sciences, such as software engineering [228], can be contrasted with the description-driven explanatory sciences where the focus is to describe, explain and possibly predict observable phenomena in the field of study [3].

The design knowledge can be captured and described in a technological rule, defined by van Aken [3] as *"a chunk of general knowledge, linking an intervention or artefact with a desired outcome or performance in a certain field of application"*. Storey et al. [204] continue and explain how a technological rule captures general knowledge mapping between a problem-solution pair. A rule can be phrased as: *"to achieve <Effect> in <Situation> apply <Intervention>"* [204].

A technological rule can in this sense be compared to what Wieringa refers to as a treatment [228] which he borrows from the health care domain - *"... it naturally suggests an artifact (medicine) interacting with a problem context (the human body) to treat a real-world problem (contribute to healing)"*. Design science research should hence consider not just the artifact independently, but also how it interacts with the problem context. The artifact can conceptually be *"any designed object in which a research contribution is embedded in the design"* [179], e.g., methods, frameworks, and algorithms [90, 228]. The artifact can thereby be seen as a container for what van Aken refers to as design knowledge.

The process for conducting design science research is a creative and iterative process carried out in design cycles [228]. A design cycle consists of three tasks: problem investigation, treatment design, and treatment validation (cf. processes suggested by Peffers et al. [179] and Hevner [89, 90]). In the problem investigation, the mission is to understand the problem and gather knowledge about its

context. Knowledge gained is then used to design a treatment which is then validated in a modeled version of the problem context. Knowledge gathered from the validation is then used to refine problem understanding and the treatment design in the next design cycle. When the validity of the treatment is deemed acceptable, the treatment is implemented, i.e., transferred and applied to a real-world problem context and evaluated (cf. technology transfer [79]). The implementation and its evaluation are not part of the design cycle, but of a larger engineering cycle, usually carried out outside of the scope of design science research projects [228]. This aligns with van Aken who emphasizes that design knowledge and technological rules should be used by the practitioners in the problem context who in turn can develop context-specific solutions.

In this thesis we use the terminology proposed by Wieringa with the difference that we refer to the artifact directly instead of the treatment. The interaction between the artifact and the problem context is described implicitly in the presentation of the artifacts. For example instead of treatment design and validation, we refer to these phases as artifact design and validation.

As highlighted in literature [204], rigor and relevance are two important criteria in design science research. A risk is that one of the criteria will suffer to the benefit if the other. For example, if a study is too focused on rigor and the theoretical contributions, design of the artifact and relevance for practitioners may suffer. Conversely, if the research design is too pragmatic and flexible, rigor and quality of the artifact's underlying design knowledge may suffer. Hence, design science research needs to consider and balance both rigor and relevance when design cycles are conducted [89, 90].

2.2 The Case Study Research Method

Within a design science research project, several empirical software engineering research methods may be applied, e.g., case studies [191], experiments [233], surveys [150] and considering expert opinions [228]. In the artifact validation phase specifically, technical action research [229] may be a further option where a prototype of the artifact is field-tested in a real-world problem context. Of these, the case study is the primary method used in this thesis (see *Papers I, II, IV* and *V*). van Aken mentions the multiple case study as a typical method to study and test technological rules [3]. Design knowledge generated from one case can be tested and refined in another.

Following the definition by Runeson et al. [191], a case study investigates an instance of a contemporary software engineering phenomenon within a real-world context [191]. The method can offer in-depth knowledge and understanding of the phenomena in how and why it occurs [50]. In the problem investigation phase, the case study will focus on observing and investigating the phenomenon without intervening [228], i.e., maintaining either an exploratory, descriptive or explanatory purpose [191]. Case studies with an improving purpose, similar to action research,

may also be applied in the latter phase of a design cycle, i.e., the artifact validation [191, 228]. In this phase, the case study will focus on the introduction of an artifact prototype into a real-world problem context and study the effects that arise, and whether these are in line with the stakeholders' expectations.

A case study can be either inductive or deductive in its investigation [191]. The former implies that theory is derived from the observations, while in the latter, research starts from a theory with a set of hypotheses which are either confirmed or rejected in the analysis.

2.3 Research Design

This thesis presents five papers referred to as *Papers I-V* as presented in Fig. 1 and Table 1. Depending on the research objective, the papers can have an exploratory or improving focus. *Paper I* investigates the problem context and creates a knowledge base for *Papers II-V* by exploring how a software-intensive organization engages with OSS communities' RE and development processes from an OI perspective through a case study on Sony Mobile. The paper rendered in the definition of *RG1* of this thesis, which was addressed through a continued research collaboration with Sony Mobile in *Paper II*. The collaboration provided an opportunity to get in-depth knowledge on their ways of working with contribution strategies and designing a first artifact that may be used to develop such strategies.

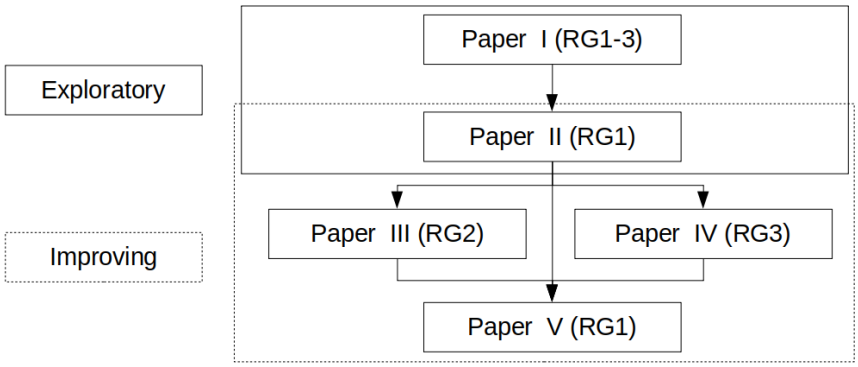


Figure 1: Overview of *Papers I-V*, and how they relate to *RG1-3* and each other.

Based on the new understanding of the problem context, *RG2-3* of this thesis were defined. *RG2* is addressed in *Paper III* where a framework for developing community strategies is presented based on an iterative interview survey. *Paper IV* proposes a method for analyzing stakeholders' influence on an OSS community's RE process in response to *RG3*. The method is based on a social network analysis approach used in earlier work [132] and a literature review, and is demonstrated through a case study of the Apache Hadoop OSS community. Lastly, *Paper V*

Table 1: Overview of the different papers and their main objective, type of data and design cycle phases represented.

Paper	RG	Research method	Main objective	Type of data	Design cycle phases
I	1-3	Single-case study	Exploratory	Quantitative and Qualitative	Problem investigation
II	1	Single-case study	Improving	Quantitative and Qualitative	Problem investigation, artifact design, artifact validation
III	2	Interview survey	Improving	Qualitative	Problem investigation, artifact design, artifact validation
IV	3	Single-case study	Improving	Quantitative	Problem investigation, artifact design, artifact validation
V	1	Multiple-case study	Improving	Qualitative	Problem investigation, artifact design, artifact validation

advances the results from *Paper II* and addresses *RG1* by presenting a framework for developing contribution strategies and setting up a contribution process based on a multi-case study with two private and one public organization. Below we present the research design of each paper in detail.

Paper I

Title: Open Innovation using Open Source Tools: A Case Study at Sony Mobile

Paper I addresses *RG1-3* through an exploratory case study [191] at Sony Mobile and its Tools department. The Tools department consists of about 15 engineers which develop and maintain the development infrastructure for Sony Mobile's product development teams. One such infrastructure includs the continuous intergration tool-chain. Units of analysis for this study were the Jenkins and Gerrit OSS, which both constituts pivotal parts of the this tool-chain. Jenkins[4] is a build server, and Gerrit[5] a tool enabling peer-review of code commits.

From an OI perspective (see Fig. 2), the Tools department constituted the focal point of Sony Mobile through which the interactions with the Jenkins and Gerrit OSS communities was performed. A case-study protocol was created and maintained during the process of conducting the study. The study started with mining

[4]https://jenkins.io/
[5]https://www.gerritcodereview.com/

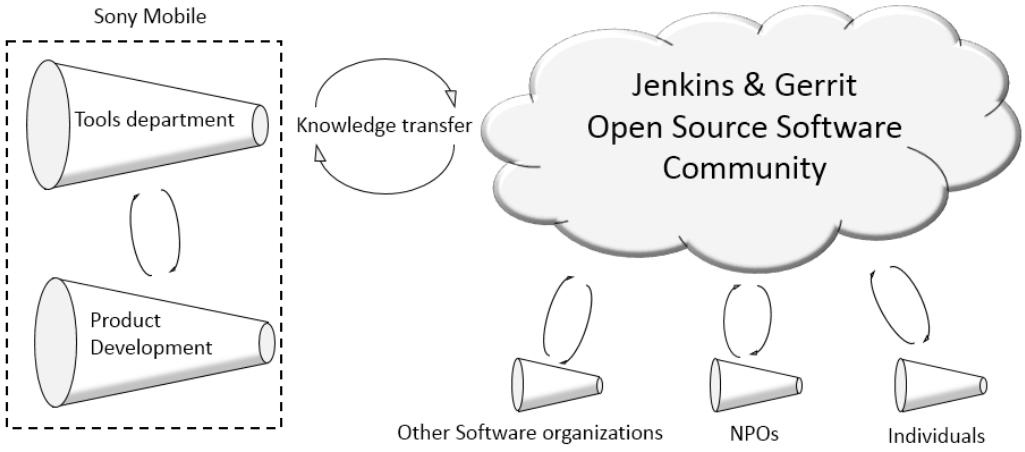


Figure 2: The Jenkins and Gerrit OSS communities surrounded by Sony Mobile and other members. Arrows represent knowledge transfer in and out of the community members such as other software organizations, non profit organizations (NPO) and individuals, which in turn are illustrated by funnels, commonly used in OI literature [30].

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 an organizational level, and type and distribution of commits. Semi-structured interviews were conducted with three developers at Sony Mobile that was 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 [36]. The output from the coding processes was communicated and validated with interviewees.

Paper II

Title: Motivating the Contributions: An Open Innovation perspective on What to Share as Open Source Software

Paper II addresses *RG1* using a design science research approach [90] with a case study of Sony Mobile. Sony Mobile is a multinational organization with roughly 5,000 employees, developing mobile phones and tablets. The studied branch is focused on developing Android based phones and tablets and has 1600 employees, of which 900 are directly involved in software development. Sony Mobile develops software using agile methodologies and uses software product

line management with a database of more than 20,000 features suggested or implemented across all product lines [180]. Sony Mobile is a mature user of OSS with involvement in several OSS projects. Their existing processes for managing contribution strategies and compliance issues is centrally managed by an internal group referred to as their OSS governance board [157].

Iterative design cycles included informal consultations with four experts at Sony Mobile who were involved in the decision-making regarding the contribution process to OSS communities. Internal documentation of the contribution process and policies was also used and analyzed. A Contribution Acceptance Process (CAP) model was designed based on a purchasing and sourcing model proposed by Kraljic [116]. 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. The CAP model was validated on three case organizations in terms of applicability and usability.

Paper III

Title: A Community Strategy Framework – How to obtain Influence on Requirements in Meritocratic Open Source Software Communities?

Paper III addresses *RG2* using a design science research approach [228]. The problem investigation phase included a series of ten semi-structured interviews with industry professionals to explore the problem context. The interview transcripts were coded with an inductive approach resulting in the first design of a Community Strategy Framework (CSF). To validate and refine the design, seven interviews were conducted, where the interviewees were presented with the CSF and asked questions regarding its completeness and correctness. To evaluate the applicability and utility of CSF [90], in one of these interviews, the framework was also applied to a fictitious example based on an earlier reported case study [157]. As the last step, a case validation was conducted by interviewing four industry professionals from a software-intensive organization engaged in multiple OSS communities. Questions focused on the validity of CSF in the context of the organization's community engagements.

In total, we conducted 21 interviews with 18 industry professionals from 12 different software-intensive organizations. The interviewees all held positions with responsibilities relevant to understanding how their respective firms work and engage with OSS communities. Role titles included OSS Program Officer, Community manager and OSS Strategist. Interviewees were selected based on convenience sampling. In their respective organizations, OSS was used in tool and infrastructure setups, as well as in software- and hardware-based products. 9 out of the 12 organization had a size of 250 employees or more.

Paper IV

Title: A Method for Analyzing Stakeholders' Influence on an Open Source Software Ecosystem's Requirements Engineering Process

Paper IV addresses *RG3* using a design science research approach [228]. The problem investigation phase consisted of the earlier reported case study the Apache Hadoop OSS community [132] and a review of literature related to OSS RE, stakeholder theory and analysis, and social network analysis. A Stakeholder Influence Analysis (SIA) method was designed by conceptualizing the research approach used in the aforementioned case study [132] and considering the identified literature. SIA was demonstrated in terms of applicability and utility through an extension of the earlier reported case study on the Apache Hadoop OSS community by also considering comments made by stakeholders to common issues. Hence, both patch and comments-based networks could be created. The stakeholders' interactions were studied based on their contribution of patches to and comments on issues included in releases R2.2.0 (2013-10-15) to R2.7.1 (2015-07-06). These issues were mined from the community's JIRA issue tracker by implementing a crawler. Directed affiliation-networks were created where two stakeholders (represented as nodes) were connected with edges if they had both contributed a patch to or commented on a common issue. These edges were weighed based on the size of each stakeholder's contribution relative to the other in terms of net changed lines of code or the relative number of comments. Developers' organizational affiliation was determined based on email-domain analysis complemented with a qualitative heuristical analysis of electronic sources [18, 78]. The patch and comments-based networks were then analyzed based on a fictive case. The extended case study helped to refine the design of SIA.

Paper V

Title: A Contribution Management Framework – What to share as Open Source Software?

Paper V addresses *RG1* using a design science research approach [228] with three design cycles. Through collaboration with Sony Mobile, the first design cycle presented by Linåker et al. in earlier work [128] designed a Contribution Acceptance Process (CAP) model for creating contribution strategies [128]. The second and third design cycles cover a multiple-case study involving three case organizations (CaseOrg1-3) and are presented in this study.

CaseOrg1 is a US-based media and technology company providing video, high-speed internet, smart home and voice services. They have 1000+ employees and develop their own software in order to enable and deliver their services to the customers. Having been passive consumers before 2006, they became active contributors starting in 2006. Since, they have released a number of OSS projects

and are active contributors in several others, as well as members of a number of OSS foundations.

CaseOrg2 is a European-based hardware electronics manufacturer serving both business and private customers. They have 1000+ employees and develop their own software in order to enable and deliver their services to the customers. This study has focused on its Tools department which develops and maintains development tools and infrastructure projects used by the organization's product development teams. A majority of the tools and infrastructure projects are OSS-based with active engagement from the Tools department in their respective communities. The active engagement includes continuous contributions of features and plugins to existing OSS communities as well as the release of new OSS projects.

The Swedish Public Employment Service[6] makes up the third case organization but is referenced to as CaseOrg3 for consistency. CaseOrg3 is a public sector agency in Sweden with the main goal to facilitate and enable matching between job-seekers and employers on the Swedish labor market. The organization has 10 000+ employees of which 600 are employed within their IT division. The focus of this study is on a department within the IT division which aims to create a platform[7] on which private actors can build complementary products and services for matching job-seekers and employers. The platform, consisting of OSS projects and Open Data sources, is intended to help CaseOrg3 move from the role of being a service provider to becoming a service enabler, and helping the platform's ecosystem members to collaborate and co-create, potentially resulting in accelerated innovation and a more efficient job-matching on the Swedish labor market.

In the second design cycle, a questionnaire was developed based on constructs from the CAP model and used in six semi-structured interviews at CaseOrg1. A first draft of the CoMn framework was the created based an inductive coding of the interview transcripts. Archival analysis of contribution request forms from seven organizations were also performed and cross-mapped against the first draft of the framework. In the third design cycle, a new questionnaire was developed based on results from the previous cycle and used in five and nine interviews respectively in CaseOrg2 and CaseOrg3. Using the coding schema from the second design cycle, transcripts from CaseOrg2 was first coded and followed by the transcripts from CaseOrg3. Transcripts from all case organizations were then reiterated, and a final coding scheme assembled. The revised version of the CoMn framework was then presented to interviewees from the three case organizations who were asked questions regarding the framework's completeness and correctness, which resulted in minor modifications.

[6] See: https://arbetsformedlingen.se/
[7] See: https://jobtechdev.se/

3 Background and Related Work

First, in subsection 3.1, we give a background on how RE in OSS communities may function and the need for influence to affect, e.g., selection and prioritization of requirements. Second, in subsection 3.2, we present the role of governance and authority structure in an OSS community in regards to how influence may be gained. Third, in subsection 3.3, we continue on the role of governance and authority structure but focus on cases where a community is institutionalized through a foundation. Fourth, in subsection 3.4, we present earlier work on how organizations can gain influence on the RE process in meritocratic OSS communities. Lastly, in subsection 3.5 and 3.6, we focus on previous work describing the rationale for an organization to share software as OSS and engage with an OSS community, both concerning intangible benefits and more commercial reasons respectively, and end with a summary in subsection 3.7.

3.1 Open Source Software Requirements Engineering

In OSS communities, requirements practices are often informal and overlapping [27, 192]. Ernst and Murphy refer to it as a lightweight and evolutionary process of requirements refinement and labels it Just-In-Time (JIT) requirements, compared to the more traditional upfront requirements characterized by heavy processes and tool support [54]. Further, Alspaugh and Scacchi contrast how OSS RE moves 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 [4].

In traditional RE, requirements are elicited from the stakeholders using suitable elicitation techniques compared to OSS RE where requirements are commonly asserted by the OSS project's stakeholders, i.e., developers and users. This is done through transparent discussions and suggestions and often together with prototypes or proof-of-concepts [4, 74]. The assertion may also be done post-hoc, simultaneously as the requirement realization [54, 74]. These assertions are specified and managed in what Scacchi refers to as informalisms [192], e.g. reports in an issue tracker, messages in a mailing list, or commits in a version control system. Through continual social interaction facilitated by the infrastructure persisting the informalisms, requirements are further enriched and validated [27, 54, 74, 209]. This interaction can however also occur centralized in "off-line" events such as conferences, meet-ups, and hackathons [38, 157, 202].

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 [2, 74, 120, 167]. This hierarchy between the roles in OSS communities is often depicted with the help of an onion model [161]. In its multi-layered construction, central and leadership roles can be found among the core layers, while

the passive users can be found in the outer ones (cf. Core-Periphery Model [103]). The structure implies that the further out a community member is, the less direct influence and knowledge the person has over the project's state and direction [101]. Furthermore, what roles that exist in a community, specifically regarding leadership, may differ between communities. Some may, for example, have a project lead as with Linus Torvalds in the Linux kernel community, while some may have a core team of entrusted members as in the PostgreSQL community [161].

Migration between layers can be fluid and agile depending on the project, e.g., community members can move between multiple layers, or be recruited into one, bypassing outer ones [101]. This migration further depends on the type of governance in the community.

3.2 Governance in Open Source Software Communities

De Laat [46] describes OSS governance as different configurations, primarily based on the authority structure, i.e., the way that authority is established, distributed, and exercised, either through autocratic or democratic principles. In the former, leadership is centralized and top-down, while in the latter it is decentralized and bottom-up. Building on this distinction, De Noni et al. [47] refines the two configurations further as presented in Fig 3. Concerning communities with autocratic tendencies, they differentiate between sponsor-based and tolerant dictator-based communities. In the former, leadership is centered around the sponsoring organization(s), while in the latter it is centered around a single project leader (tolerant dictator). In regards to communities with democratic tendencies, De Noni et al. [47] separates open-source-based and collective communities. In open-source based communities leadership is characterized as institutionalized, democratic, and distributed, often inside the walls of a foundation. In collective communities, leadership is seen as collective, meritocratic, and distributed.

<table>
<tr><th colspan="4">Governance and Authority Structure Concepts and Relations</th><th>Reference</th></tr>
<tr><td colspan="2">Autocratic communities</td><td colspan="2">Democratic communities</td><td>de Laat (2007)</td></tr>
<tr><td>Sponsor-based communities</td><td>Tolerant dictator-based communities</td><td>Open-source-based communities</td><td>Collective communities</td><td>De Noni et al. (2013)</td></tr>
<tr><td>Commercial OSS projects</td><td colspan="3">Community OSS projects</td><td>Capra and Wasserman (2008)</td></tr>
<tr><td>Single-vendor OSS projects</td><td colspan="3">Multivendor OSS projects</td><td>Schaarschmidt et al. (2015)</td></tr>
</table>

Figure 3: Overview of governance and authority structures in OSS projects. Adopted from *Paper III* in this thesis.

Capra and Wasserman [26] makes a distinction between commercial and community OSS. In the former, the OSS project is owned and managed by a single organization [186], i.e., a special case of sponsor-based communities [47]. In the latter, the community is owned and managed by the community, which may include one or more organizations, also aligning with the community-managed governance model as described by O'Mahony [172]. Schaarschmidt et al. [194] further label these types of projects as single-vendor projects and multivendor projects respectively.

Even with the categorizations of OSS governance models and their authority structures shown in Fig 3, other research shows that the picture can be more blurry. According to the literature review by Shaikh and Henfridsson [198], research has been consistent in describing how communities can only have one authority structure (with one notable exception [75]). Even though a community can evolve its authority structure in hybrid forms with time, a single authority structure will result in the end [174]. However, based on their view of a duality between governance and coordination, Shaikh and Henfridsson [198] move to suggest that multiple forms of authority structures can co-exist in parallel, each embedded in and operationalized by a coordination process. These coordination processes can integrate, and evolve together within a community, of which some may pass out with time and be replaced by others. In their longitudinal analysis of the Linux kernel community, they identified a varying mix of autocratic and oligarchic structures, but also semi-autonomous governing in terms of the different sub-modules. Meritocracy was continuously present through the analysis. Hence, even tolerant dictator-based communities can show traits of a community-managed [172] and meritocratic [46] governance model.

Although literature lists a number of them, meritocracy may be considered one of the more common authority structures, or type of governance in OSS communities (e.g., [24, 58, 161, 166, 171, 193]). Based on merit and the earning of trust and status in the community, individuals are granted further responsibility and authority [58]. Merit correlates to the quality and quantity of the individual's contributions [74, 206]. A common assumption is that these contributions are limited to technical code contributions, however, as is shown by Eckhardt et al. [51], this can be a simplification. Considering the onion model [161], several paths are depending on the type of role an individual possesses. Proven coordination and leadership skills are aspects that may be considered [101, 174], but not obviously captured in code commits. As pointed out by O'Mahony and Ferraro [174] in their study of the Debian community, *"Any examination of meritocracy must develop a context-specific understanding of how merit is conceptualized"*.

3.3 Open Source Software Foundations

As identified by O'Mahony [171], there is a tendency that as projects grow larger and attract more attention from organizations and other types of organizations, the

communities become institutionalized through the creation of non-profit software foundations, either standalone or embedded in an existing (e.g., the Eclipse or Linux foundations). By moving ownership of all copyright to a foundation, a legal shell is created around the community that can manage potential legal and patent issues, removing any liability from the community members [46, 133, 183]. A second reason is that it creates a single entity that can speak on behalf of the community and uphold its assets, e.g., copyrights, brand name, and trademarks [133, 183].

Membership in the community and foundation often overlap meaning that community members are eligible to vote in elections for committees and boards in the foundation. There are however cases as in the Linux Foundation where foundation membership is reserved to external organizations who join by paying membership fees [46, 133].

Organizations with interest in a community may have different reasons to advocate for the creation of a foundation. As O'Mahony [171] reports, in the case of the Apache Software Foundation, organizations primarily wanted an entity that they could make secure transactions to in terms of technical contributions as well as financial support and hardware. In the case of the GNOME foundation, on the other hand, organizations were more interested in using the foundation to get a more significant influence on the decision making in the GNOME community. As a return for the organizations' sponsorship and support, they are offered seats in on relevant boards and committees where they influence the project indirectly, depending on the power of the foundation in terms of technical decision making [24, 173]. In the case of GNOME, organizations were given seats on an Advisory Board without any formal power on the development of the project [171].

A related observation made by de Laat [46] is that as foundations grow more critical, the barrier of non-interference between a foundation and a community may come to be erased [46]. The two may instead become more intertwined, with the project becoming a mirrored reflection of an organization-governed pooled R&D/product development project [222].

3.4 Influencing Open Source Software Requirements Engineering

The members of an OSS community all have their motives for participating, social or economic [124, 183]. It may, therefore, be considered a challenge for organizations to align their internal agenda with that of the community [39, 173, 194]. A decision to add functionality may require consensus in the community and approval by the community leadership depending on the type of governance. Being too aggressive with one's agenda may have an adverse effect and result in the functionality being blocked [2].

Dahlander et al. [38] differentiate how organizations can adapt their relationship with an OSS community based on the level of influence needed. On a continuum scale, a relationship can be characterized as parasitic, commensalistic or

symbiotic. In the parasitic approach, the organization takes without giving back, by some referred to as a "free-rider". In the commensalistic approach, the organization contributes back when motivated, but focus on internal development. In the symbiotic approach, the organization also sees to the best of the community, working to align internal and external development. The alignment is created through working as peers, and building status and recognition inside the community [40].

To build a symbiotic relationship, organizations should first understand and learn to respect the needs, norms, and structure of the community [2, 24, 38, 40, 165], a form of "good citizenship" [173]. If there is a foundation encapsulating the OSS community, organizations may have the option to gain influence through membership or sponsorship [171,173], or in other ways supporting the foundation, e.g., by supporting development with infrastructure [38], or general subject matter expertise [24]. In return, they may receive seats at relevant boards and committees through which they can make their voice heard [24, 171]. Foundations and similar boundary organizations between organizations and an OSS community are often limited to managing the technical direction of an OSS project [173].

A more direct and general approach to the control of code contributions is by having "a man on the inside", letting employees engage with the community [40, 87, 165, 166, 173, 194]. An alternative is to contract members of the community directly to have them work on matters of importance to the organization [39, 74, 173, 185, 194]. Through their engagement, these sponsored community members can take part in the RE processes by participating in discussions and providing both technical and non-technical contributions and support [24, 157]. This work may take place both online and offline, because being visible and active on both ends is essential [157, 174, 194, 202].

According to Schaarschmidt et al. [194], this approach to gain influence through active engagement can be divided into two categories, control by leadership, and resource deployment control. In the former, influence is obtained by having employees in leadership positions of a community. In the later, influence is gained by having employees work in the community and infusing the community with the organization's norms and values.

3.5 Intangible Benefits for Sharing Open Source Software

Sharivar et al. [197] distinct between two types of benefits that may come out from from sharing software as OSS; tangible revenues and intangible benefits. Tangible revenues are generated from sales in cases where an organization offers complementary products and services based on the OSS [6, 197, 200] while intangible benefits are gained as an effect of sharing software OSS. Reasons for why an organization would choose to share software as OSS does not have to be limited to one or the other [6]. Below we focus on the intangible benefits, which has been systematically surveyed in literature to different extents [84, 93, 159, 197]. We categorize the benefits into four different themes.

A first theme and type of rationale for sharing software as OSS may be to gain influence on the development direction of the community by participating in the development and maintaining a symbiotic relationship [24, 38, 40, 166, 194, 206], as also noted in Section 3.4. This may help steer the community including competitors and to manage potentially conflicting agendas [138, 159, 194, 225]. Influence may also come implicitly when an organization's project or a feature is released and accepted as a standard solution, either within an existing or as a new community [157]. If contributed to an existing community, other organizations will either have to accept and adapt, maintain internal forks of their own solutions, or attempt to contribute their solutions in competition with the solution already established within a community [128]. If released as a new community and traction is gained, it can potentially become a new standard or compete with existing [86, 98, 134, 219], and create a surrounding community with complements from other organizations [220].

Another common theme in literature concerns cost-saving aspects [6, 159]. By extending the resource-base [39] and agreeing on a common standard [222], organizations can share the maintenance and quality assurance, accelerate the development and potentially decrease their time-to-release and market [86, 91, 134, 157, 205]. By freeing up internal resources, they can focus on more value-adding activities [134, 157, 212]. On the contrary, by adopting a less symbiotic relationship to the OSS community [38], an organization will have to maintain an internal branch of the OSS project which may become costly depending on the number of modifications that need to be applied to new releases of the OSS project [213,231]. Hence, to attain these potential benefits, an active engagement and symbiotic relationship may be needed with the OSS community [29, 40]

A third common theme concerns innovation aspects [6,159], which can be both product and process-oriented [158]. By opening up the innovation process [32] and "pooling" the R&D/product development [222], organizations get access to an external workforce which may bring increased knowledge sharing [160] and innovation at a lower cost [205, 234]. However, this external workforce should rather be seen as a complement rather than a substitute for internal knowledge and development [39, 202]. Munir et al. [159] describe it as a catalyst for ideas that may help organizations in broadening their offerings. Hence, an organization may question how much of its internal R&D and innovation process it should outsource to a community [2].

A fourth theme can be tied to improved reputation [159, 213]. By creating a community or contributing to an existing one, an organization can create a marketing channel both towards (potential) customers, as well as future employees [39, 86, 185, 205]. The improved reputation can turn into a competitive advantage [88] and legitimize the use of the OSS from a public perspective [40]. An organization's customers are offered an opportunity to avoid vendor-lockin, and the ability to customize the software to internal needs [157].

3.6 Open Source Software in the Business Model

To consider the tangible revenues as described by Sharivar et al. [197], the Business Model (BM) concept provides a potential framework as it describes how the pieces of an organization's business fit together [140]. This puzzle explains how an organization creates and delivers value to its customers, and then captures the value [207]. However, even though research is converging, there is still some heterogeneity in the definition of what a BM is, and of what components it consists [45, 230]. As a consequence, there is also confusion in how OSS connects to a BM with multiple and overlapping definitions (e.g., [32, 197, 216]).

Below follows an attempt in categorizing how OSS may be used as a complement to the value proposition of an organization, either as part of or enabler for the creation of products or services [124]. As reported in literature, these can often overlap and be combined [32, 39]

Professional services and consulting: An organization builds its value proposition on the knowledge and absorptive capacity they possess in relation to an OSS project and become product specialists. Services such as consulting, training, support and customization of the OSS project are offered based on a customer's needs [6, 31, 85, 113, 184, 216]. Red Hat[8], and IBM[9], two large software product organizations which base their products on OSS, both sell complementary services such as training and consulting [31, 156].

Subscription and support: An organization offers a stable and maintained version of a specific OSS project to a customer. During a subscription period, a customer is provided with updates, upgrades and support [31, 85, 113, 184, 216]. Red Hat, again a prime example, offers its customers maintained versions of popular OSS projects such as OpenStack[10], Kubernetes[11] and Fedora[12] [31, 156].

Open core and proprietary extensions: An organization uses one or more OSS projects as an open core which is used in a commercial version with enterprise features only available under a proprietary license [184]. The enterprise features may also be available as extensions and plugins which can be integrated and used on top of the intended OSS project [32,216]. Hortonworks[13], a large software vendor, uses an open core model by selling its distribution, the Hortonworks Data Platform, of the Apache Hadoop[14] project bundled together with complementary OSS projects such as Apache Hive[15] and Apache Spark [132]. MySQL[16], a database vendor, on the other hand, is focused on selling proprietary extensions to its database which is available as OSS [32, 39, 113].

[8] See: https://www.redhat.com/en
[9] See: https://www.ibm.com/
[10] See: https://www.openstack.org/
[11] See: https://kubernetes.io/
[12] See: https://getfedora.org/
[13] See: https://hortonworks.com/products/
[14] See: https://hadoop.apache.org/
[15] See: https://hive.apache.org/
[16] See: https://www.mysql.com/products/

Dual licensing: An organization has released its product as an OSS project under a copy-left license (usually GPL2, GPL3 or AGPL) [170]. This requires a user of the project to contribute any changes back that the user makes to the project, but only if the user re-distributes the project outside of its legal entity. To bypass this requirement, a customer can pay for a proprietary license in which there are no copy-left requirements [113]. This is possible as the organization owns the full copyright to the OSS project and thereby determines what license apply [186]. MySQL and their database software is a well-reported example of this model [30, 39, 113, 216].

Hardware device enabler: An organization embeds an OSS project into a hardware device and thereby enabling their value proposition which may be to sell hardware devices or services related to the devices [32, 85, 113]. Sony Mobile, a consumer electronics organization, uses the Android Open Source Project[17] to enable selling of their mobile handset devices [128]. Cable providers like Comcast, on the other hand, use the Reference Design Kit[18] project in their set-top boxes which allows Comcast to sell cable services to their customers.

Online service enabler: An organization uses an OSS project as the basis for value propositions including online services [85]. These services may, for example, be hosted instances of the OSS project or providing the OSS project using a Software-as-a-Service (SaaS) model [113]. The customer is offered a current, stable and maintained an instance of the OSS project without having to install it in on its premises. Amazon Web Services[19], a cloud service provider, offers customers to run OSS projects such as Kubenetes and OpenStack on their cloud platform. As another example, Wordpress[20], an OSS content management system vendor, offers customers their OSS project through a SaaS-model.

Complement enabler: An organization develops and sells hardware and software products and services in adjacent and independent layers of a software stack, on top of an OSS-based layer [183,184]. As reported by Koenig [113], by supporting and optimizing Linux to work on their hardware platforms, Oracle can offer customers a solution at a lower cost, and equal or better performance compared to proprietary solutions and avoiding vendor lock-in at the same time.

Tools and infrastructure: An organization uses OSS as part of internal tools and infrastructure that enables the development and delivery of the organization's product or service [113]. The OSS is developed based on internal needs with an indirect, rather than direct, connection to the organization's value proposition [32, 216]. As an example, Sony Mobile leverages OSS projects such as Jenkins and Gerrit in their internal continuous integration tool-chain [157]. This enables Sony Mobile to tailor the tools based on internal requirements, which can result in higher quality products with a faster time-to-market.

[17] See: https://source.android.com/
[18] See: https://rdkcentral.com/
[19] See: https://aws.amazon.com/
[20] See: https://wordpress.com/

3.7 Summary of Background and Related Work

For an organization engaged in an OSS community, it is pivotal to consider how much influence that is needed on the community's RE process for the organization to maintain its agenda [24,38,39,194]. In a community, the focal organization is a stakeholder among many, all with their motives for engaging [132, 159, 173]. The RE process in these communities can often be described as informal and decentralized with a just-in-time mentality [4,54,193]. Also considering the variety in governance processes [46, 47, 174, 198], aligning external and internal RE along with building up influence needed can be a big challenge for firms [39, 159], especially as there is no contractual arrangement to refer to [2,38]. In meritocratic OSS communities, an organization can gain influence by proving merit and earning trust and status among its peers through an active and symbiotic engagement [38, 58, 206]. Technical contributions are often cited as a way to prove merit although this may be a simplification [51, 174].

The ways in how OSS can help an organization to create and capture value from a business model perspective are manys [6, 31, 85, 113, 184, 216], as are the number of more intangible benefits such as cost-savings, improved innovation and reputation, but also influence on a community's RE process [84,93,159,197]. Evaluating how an OSS project and its community provide such value, both monetary and non-monetary, may help an organization prioritize which communities that really matter [129]. Such evaluation may also provide input to decision-makers on questions and requests of contributing internally developed software as OSS [128]. These matters are explicated in *RG1-3* which this thesis aims to address with the included *Papers I-V*.

4 Results

In this section, results of each paper as listed in Table 1 are presented with particular focus on research goals stated in section 1 with the corresponding research methodology as presented in section 2.

4.1 Paper I: RG1-3

Title: Open Innovation using Open Source Tools: A Case Study at Sony Mobile

In *Paper I* we explore how a software-intensive organization engages with OSS communities' RE and development processes from an OI perspective. Its results provide a knowledge base for the following solution-oriented papers *II-V*.

The study shows that the main reason for Sony Mobile to "open up" was a general shift towards adopting OSS as a platform for the organization's products. As a 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 OSS communities 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. A main enabler for this openness was that tools and infrastructure such as Jenkins and Gerrit were considered as non-competitive and non-pecuniary.

The importance of having an active presence and influence in a community was exemplified in the reported example of the Gerrit-Trigger-plugin. The plugin was first considered a source of competitive edge, but when the developers at Sony Mobile heard about an alternative solution they were quick to share the plugin as OSS. As reported in literature [231] and in *Paper V*, this is a trade-off between keeping a competitive edge and having to maintain the internal solution or adapt to the alternative solution. By being first to contribute a solution that gets accepted, Sony Mobile got a first-mover advantage as other organizations were faced with the decision to adapt, compete, or continue with their own solution.

The Tools department's RE process that interfaced the OSS communities was informal and managed internally in an agile manner through a combination of Scrum and Kanban methods. Prioritization of issues in the OSS communities 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 communities included new and improved functionality for the two tools, as well as less maintenance and shorter internal release cycles.

The 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 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 communities and adaption in development practices have had the effect of introducing an Inner-source initiative to Sony Mobile.

As the study focused on tools and infrastructure OSS, it was pointed out that future work should also investigate the rationale for sharing proprietary software as OSS. This led to the definition of *RG1* of this thesis.

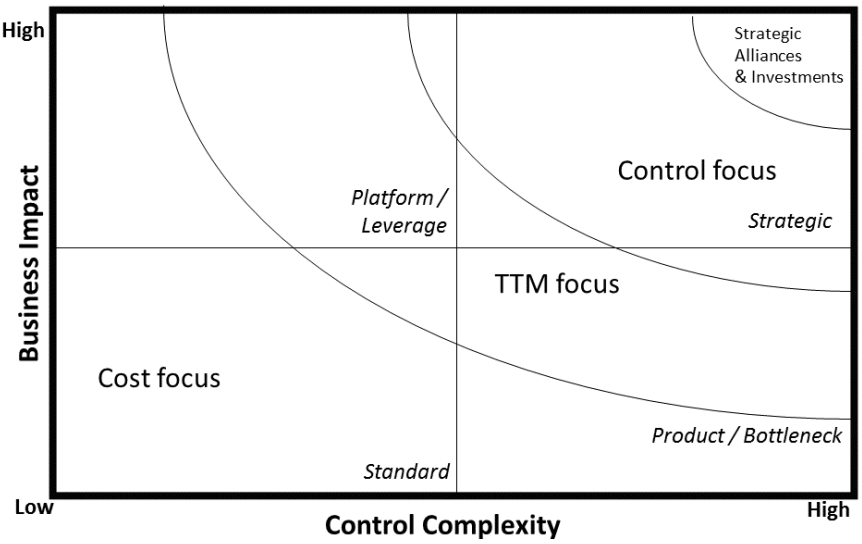


Figure 4: The Contribution Acceptance Process (CAP) model as presented in *Paper II*. Based on how a software artifact is valued in regards to its business impact and control complexity, a contribution strategy is elicited pending the artifact's placement on the grid.

4.2 Paper II: RG1

Title: Motivating the Contributions: An Open Innovation perspective on What to Share as Open Source Software

In *Paper II* we propose a Contribution Acceptance Process (CAP) model to guide organizations in developing contribution strategies for internally developed software artifacts that align with an organization's internal product strategies and planning. The study uses an earlier definition of a contribution strategy (compare to Section 1), adopted from Wnuk et al. [231] that focused on *what* should be contributed back to an OSS community and *when*.

From the functional perspective, the 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. For example strategic

artifacts are those with a high business impact and control complexity. These are differential and makes up a competitive edge for the organization. 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 screening process may, therefore, be needed for these artifacts.

A user of the CAP model also needs to consider the commoditization of software artifacts and how the artifacts deprecate in value and differentiation over time. Considering the importance of contributing certain solutions before others, the CAP model should be used in an iterative process as an artifact may move between from a strategic to a commodity state and in different paces.

Recommendations to either contribute to existing communities or create new communities vary between the contribution strategies that the CAP model applies to the four different types of software artifacts. The recommendations recognize that there is a substantial cost connected to creating a new community. When there is a need for control, and the organization does not have the necessary influence in a suitable target community, a new community may be preferred. This may also be the case when the interest in existing communities is limited and when a smaller industry consortium is more suitable. In other cases, it is more preferable to contribute to existing communities as far as possible.

To support the operationalization of the CAP model, we designed an information meta-model. The meta-model presents how a series of software artifact repositories may be set up 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.

Results from the artifact validation phase showed that the CAP model provide a good foundation for discussion. Feedback pointed out that the questions and scale used to value a software artifact in terms of business impact and control complexity, was found useful but in need of being tailored to the context where the CAP model is applied. A concern identified for future design cycles was not to ”over-engineer” the following versions of the CAP model. The study also points out that future work should consider the influence an organization needs in an OSS community to be able to exercise control and introduce new features as needed. The study further shows the importance of considering what other stakeholders that are present in a community and what their agendas are. This led to the definition of *RG2* and *RG3* respectively.

4.3 Paper III: RG2

Title: A Community Strategy Framework – How to obtain Influence on Requirements in Meritocratic Open Source Software Communities?

In *Paper III* we propose the Community Strategy Framework (CSF) to help organizations develop and tailor community strategies which describe if and why an organization needs influence on the RE process in a specific OSS community, and how the organization could gain it.

The Community Strategy Framework (CSF) therefore consists of two parts. The first of these contains aspects an organization should consider when assessing its need to influence the RE process in an OSS community. The second part of the CSF consists of practices an organization should consider in order to gain influence on the RE process in an OSS community with meritocratic governance or aspects thereof [198]. Figure 5 shows an overview of the CSF. An organization constructs a community strategy by firstly assessing the community of interest based on four Business Aspects (BA1-4) and four Technical Aspects (TA1-4), and secondly determine the actual need for and feasibility of gaining influence using the four Community Aspects (CA1-4). It may be that not all aspects are applicable or relevant. It may also be that one aspect may indicate a need for influence, while another may not. With this in mind, it is up to the user to consider the different aspects in relation to the community of interest, and weigh these against

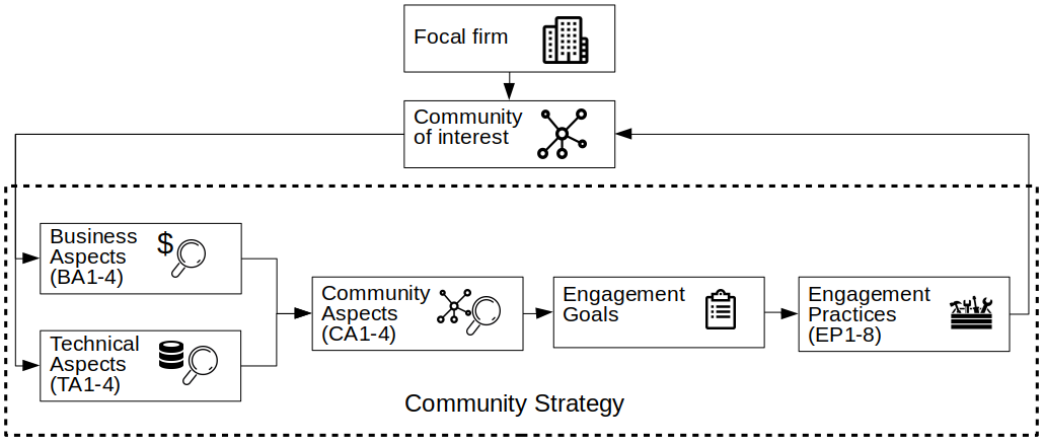


Figure 5: Overview of the Community Strategy Framework's related process. An organization first values the community of interest with the business and technical aspects and then uses the community aspects to determine the feasibility of gaining influence and potential engagement goals. Engagement goals are then decided and engagement practices chosen.

each other. The CSF should, therefore, be viewed as a support for the user to arrive at a decision on if and how much influence is needed by the organization on the RE process in the OSS community. Once such a decision has been made, the organization then formulates important engagement goals and selects which Engagement Practices (EP1-8) to apply, and finally determine how to apply them.

4.4 Paper IV: RG3

Title: A Method for Analyzing Stakeholders' Influence on an Open Source Software Ecosystem's Requirements Engineering Process

In *Paper IV* we propose the Stakeholder Influence Analysis (SIA) method which aims to help organizations involved in an OSS community to characterize the community's stakeholders according to their level of influence on the community's RE process. SIA enables organizations to see in which requirements a stakeholder holds a certain interest, and thereby create an overview of a stakeholder's agenda. This also allows organizations to understand how stakeholders invest their resources, and with whom they collaborate according to their agenda. Thus, SIA offers input to how organizations involved in OSS communities may develop their contribution and community strategies and how to act in the politics and negotiations of the community's RE process in order to align it with their internal RE process and product planning.

SIA consists of seven steps (see Fig. 6) which consider the nature in how requirements are fragmented as multiple requirement artifacts persisted in a decentralized manner in as many repositories [4, 54].

To create an overview of how stakeholders interact in the community, the framework describes how the most important repositories should be identified and

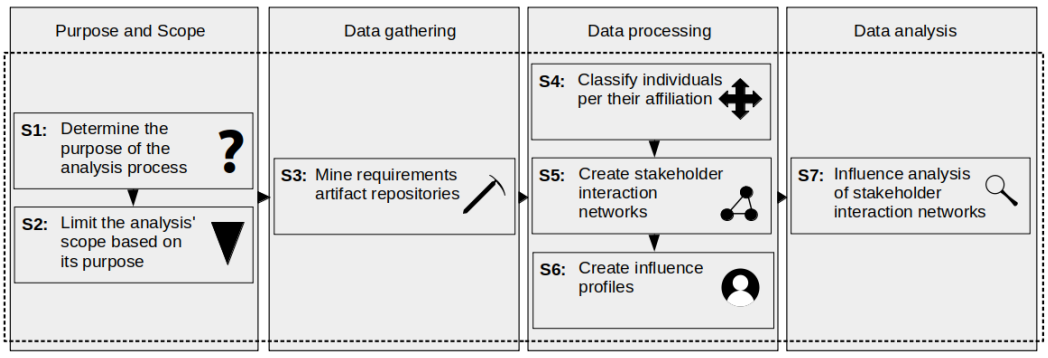


Figure 6: Overview of SIA's seven steps (S1-S7) divided in Purpose and Scope, Data gathering, processing and analysis.

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. Influence profiles can then be created by considering a series of centrality measures that have proven to be useful in characterizing the influence of stakeholders [176, 189], but also effective when analyzing an organization's participation in OSS communities [176,208] and requirement-centric stakeholder collaborations [17, 43, 142].

An influence analysis can then be performed by using a stakeholder mapping approach based on earlier work [104, 144, 162]. This may provide the focal organization with a useful tool for classifying stakeholders on a rough level to help determine what kind of engagement and relationship is needed with different stakeholders, and identify aspects that need special consideration. In this step, the quantitative picture which is generated through the networks is best complemented with qualitative insights which may be gained through observing or even taking part in the communication of the community.

Care needs to be taken to planning and executing the analysis, e.g., when defining the purpose and scope, as well as identifying organizational affiliations and creating the networks. The case study of the Apache Hadoop OSS community shows the importance to consider all requirements artifact repositories that are used within a community, and also how the different centrality metrics complement each other. Each measure can present a different social structure than another and different measures provide different perspectives on who are the most active [176]. In smaller and simpler network structures such measures may covary, while in larger and more complex networks, they may characterize actors very differently [81].

SIA does show potential in terms of applicability and utility through a proof of concept demonstration in a case study on the Apache Hadoop OSS community with a fictive setting. The study emphasizes the need for further validation in future work.

4.5 Paper V: RG1

Title: A Contribution Management Framework – What to share as Open Source Software?

In *Paper V* we propose a Contribution Management (CoMn) framework to enable software-intensive organizations to align its business goals and contribution strategies by supporting the organization in creating contribution strategy guidelines and related contribution processes for internally developed software artifacts.

While *Paper II* used the definition by Wnuk et al. [231] that focused on *what* should be contributed back to an OSS community and *when*, *Paper V* extends the definition to also consider *where*. For a specific software artifact, the "*what*" indicates if the artifact should be contributed in full or kept closed, or if certain parts

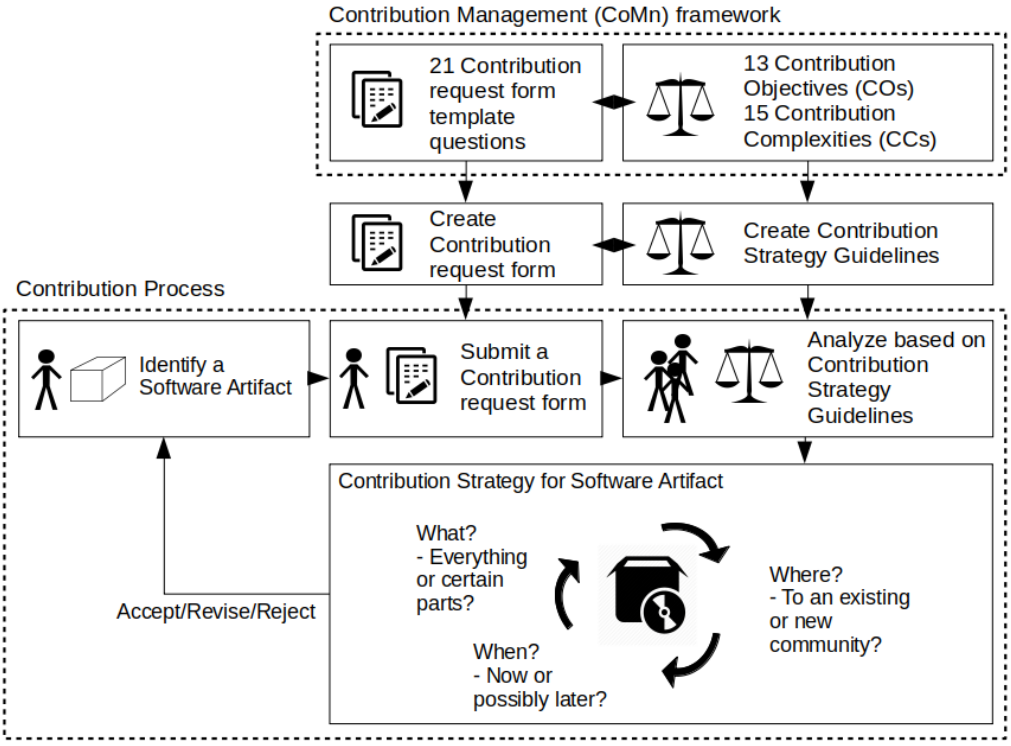


Figure 7: Overview of a contribution process, where an individual wishing to contribute a software artifact fills out a contribution request form, which is then analyzed based on an organization's contribution strategy guidelines, after which a contribution strategy is decided on. The CoMn framework can help an organization implement the contribution strategy guidelines as well as the related contribution request form.

can be contributed under certain conditions. The the "*when*" indicates when an artifact should be contributed in time. Finally, "*where*" indicates whether the artifact should be contributed to an existing OSS community or if a new community should be established.

The framework consists of a series of contribution objectives (COs) and complexities (CCs) which an organization may consider and weigh against each other when deciding on a contribution strategy for a certain software artifact. A contribution objective is defined as *a purpose for contributing a software artifact, motivated by a monetary or non-monetary benefit that is enabled or resulted directly or indirectly as a consequence the contribution*.

A contribution complexity is defined as *an aspect of or related to a software artifact that may complicate the contribution of the artifact, or imply a cost or risk as a result thereof, either directly or indirectly*.

Not all contribution objectives and complexities may be relevant for the organization in general or in certain cases. To help guide decision-makers and those making contribution requests, contribution strategy guidelines can be developed based on the objectives and complexities that are identified as generally relevant for the organization. The identified objectives and complexities can then be described in the context of the focal organization and relevant questions asked in a contribution request form when setting up a contribution process within the organization. An individual intending to file a request is then enabled to beforehand understand the rationale used by the decision makers when deciding on a contribution strategy, and thereby to provide motivated arguments why the request should be accepted. To also support the request step of the contribution process as visualized in Fig. 7, the CoMn framework also provides a series of template questions mapping to the contribution objectives and complexities that can be used for designing suitable contribution request forms.

5 Synthesis and Main Contributions

Following the recommendations by Storey et al. [204], the main contributions of this thesis are summarized in the following Technological Rules (TR) [3], related to the three research goals:

TR1: To achieve alignment between business goals and what is shared as OSS, software-intensive organizations should develop contribution strategies as well as contribution strategy guidelines and related contribution processes for internally developed software artifacts using the CAP model and the CoMn framework presented in *Papers II* and *V* respectively.

TR2: To identify OSS communities that align with business goals and achieve its internal agenda within these communities, software-intensive organizations should develop community strategies using the CSF presented in *Paper III*.

TR3: To create a contextual awareness of an OSS community's stakeholder population and input its contribution and community strategies, software-intensive organizations should analyze stakeholders' influence on the community's RE process using the SIA method presented in *Paper IV*.

The four artifacts mentioned above and presented in *Papers II-V* interplay and provide input to each other as visualized in Fig. 8.

From the perspective of a stakeholder influence analysis, knowledge about whom is present in a community, their agenda, and level of influence on the community's RE process, can provide input to the development of both contribution

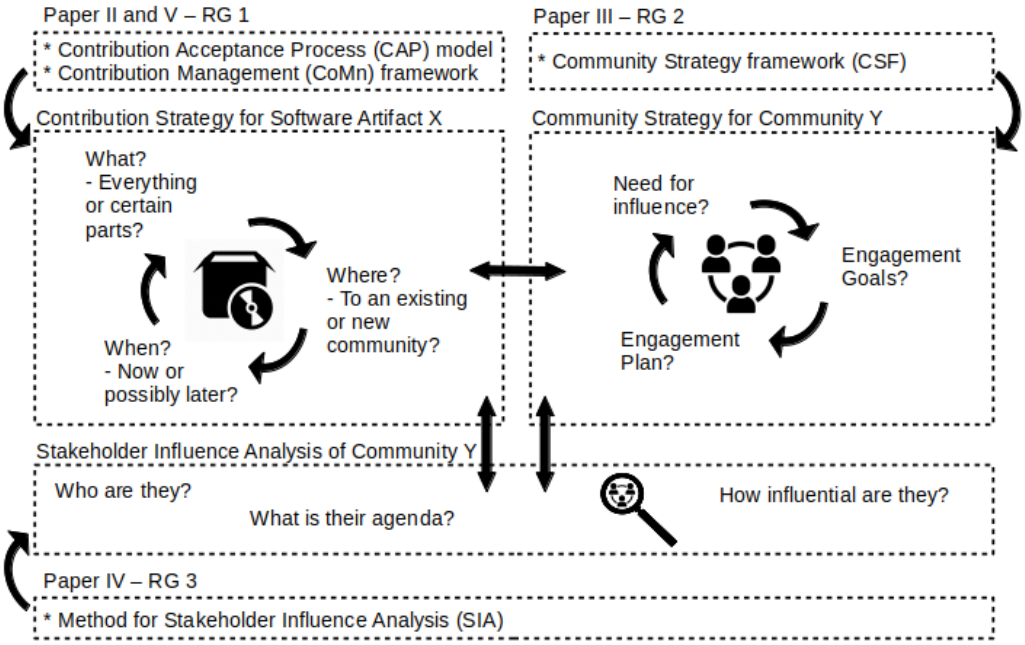


Figure 8: Overview of how the main contributions addressing *RG1-3* of this thesis interplay and complement each other.

and community strategies. Concerning contribution strategies, knowledge about competitors that are present in a community may, e.g., affect *"what"* the focal organization considers as differentiating and as a competitive edge. Such knowledge may also impact the *"when"* as the focal organization may want to keep certain software artifacts closed as long as possible, or release it as quickly as possible. The *"where"* in a contribution strategy may also be impacted, e.g, as conflicting agendas may prevent a contribution of being accepted in a specific community. This example further highlights the interplay between a stakeholder influence analysis and a community strategy, i.e., if the focal organization should invest in the community to gain the influence needed, and if this is even possible.

In terms of community strategies, these may enable the execution of contribution strategies and to maintain an influence on important software artifacts once they have been contributed. A community strategy may also help set more general contribution strategies for software artifacts that targets a certain community, e.g., in the case where the OSS project is considered as non-competitive and an important asset for the organization (as exemplified in *Paper I*). Conversely, a contribution strategy may help to identify communities where an organization need to build an influence and develop or revise a community strategy accordingly.

These dependencies, as visualized in Fig. 8, come natural as *RG2-3* evolved when initially investigating RG1 in *Paper II*. An organization interested in developing contribution or community strategies should, therefore, consider both parts as complements to each other, and also how a continuous stakeholder influence analysis in concerned communities may provide input to both types of strategies.

Below, we present a synthesis of the results from *Paper I-V* in relation to each other and their respective research goals (*RG1-3*) as presented in Fig. 1 and Table 1.

5.1 Research Goal 1

In regards to RG1, two artifacts are presented, the Contribution Acceptance Process (CAP) model (*Paper II*) and the Contribution Management (CoMn) framework (*Paper V*). Even though the intent of the two artifacts is similar there are some differences. This comes naturally as *Paper V* extends the problem investigation beyond Sony Mobile (*Paper I* and *II*) to consider three other organizations, referred to as CaseOrg1-3.

The CAP model provides a visual tool with its two-by-two matrix and step-by-step process to classify software artifacts and identify pre-defined contribution strategies that can then be adjusted. The CoMn framework extends the two dimensions of the CAP model with the two categories of contribution objectives and contribution complexities. The contribution objectives explicate different types of benefits that may be gained as a consequence of a contribution, while the contribution complexities exemplify aspects that may complicate the contribution, or in other ways imply cost or risk for the organization.

An organization may elicit those objectives and complexities relevant to them and can develop custom contribution strategy guidelines accordingly. A contribution process can then be set up with tailored contribution request forms that can help decision-makers ask the right questions, and for the person filing the request, to better motivate why the request should be accepted. The guidelines can then help to guide decision-makers in a similar manner as for the CAP model, but as a checklist when deciding on a contribution strategy for a software artifact. Based on the checklist and similar to the CAP model, examples can be created, what Kemp [107] refers to as "Do's and Dont's", which can help to guide both decision makers and those wanting to make a contribution.

Hence, while the CAP model provides an operational tool, the intention with the CoMn framework is for an organization to use it as a tool to develop their own custom contribution strategy guidelines. The two artifacts could, therefore, be seen as complements, providing context and input to each other when setting up a contribution process. The CoMn framework provides a foundation for creating the guidelines and template questions for creating contribution request forms. The CAP model provides concrete examples of contribution strategies and related software artifacts, but also a context for how the contribution strategy guidelines may be used in a proactive and reactive approach.

Adopting these approaches may help to streamline and decentralize decision-making in the contribution process. For example., when using the reactive approach and answering to more complex contribution requests, a cross-functional group with an executive mandate may have to decide what contribution strategy to proceed with, while smaller and less complex contribution requests may be managed on a lower organizational level, e.g., by the engineering managers. In the proactive approach, i.e., when using the contribution strategy guidelines in the product planning process, the guidelines can provide support for a software product manager when deciding if e.g., certain features should be released as OSS.

Considering the contribution objectives of the CoMn framework, the cost-saving and innovation-related benefits of an extended workforce and open innovation process, both confirm findings from Sony Mobile as well as literature [86, 91, 134, 157, 205]. So does the control-related benefit of creating or replacing a new standard solution within a community or industry, again considering the example of the Gerrit-Trigger-plugin, as well as the control focus highlighted in the CAP model (see also [219,222]). Objectives not found at Sony Mobile specifically but in literature, for example, connect to improved reputation [39, 86, 185, 205] within a community and the organization itself, as well as towards partners, customers and potential employees. The objectives of gathering and enabling the use of data, as well as putting price pressure on third-party vendors were neither found in Sony Mobile or reviewed literature.

Different perspectives could also be applied to the same objective. CaseOrg1-2, as well as Sony Mobile, all saw the objective about creating or replacing existing industry or community standard with the key benefits of forcing competitors to

adapt and steering the market and community development according to their own agendas. CaseOrg3 on the other hand, which is a public sector agency, saw the key benefit as improving competition and helping private actors to focus on more value-adding activities.

In regards to complexities of the CoMn framework, some alignment was found with the CAP model and literature [86, 87, 96, 212, 222, 231] combined, e.g., regarding the level of product differentiation and competitive edge, commoditization of software artifacts or the risk of giving away sensitive IPRs. Some complexities, such as the cost of the contribution and the external interest for the contribution were only implicitly considered in the CAP model, while e.g., the health of a community [214] was not considered.

Regarding the different types of organizations, CaseOrg3 did not experience the commoditization of software artifacts or the risk of giving away sensitive IPRs as complexities, while a potential ethically miss-use of the software was a concern not recognized by CaseOrg1-2. As with the objectives, different perspectives may apply to the complexities as well. CaseOrg1-2, as well as Sony Mobile, all view the level of differentiation and competitive edge of a software artifact as a risk that could hurt their own business, while CaseOrg3 views it as a risk of hurting other organizations' businesses.

Hence, one solution may not fit all. Different objectives and complexities apply for different organizations and related contextual factors. Considering the way they use and leverage OSS, all four organizations use OSS for their internal tool and infrastructure setups. In CaseOrg2, the department developing these tools are explicitly studied. Both Sony Mobile and CaseOrg1 uses OSS to enable and add value to their hardware devices. In CaseOrg1's case, OSS is also used as a basis for certain services sold to business-oriented customers. As a public agency, CaseOrg3 differs to Sony Mobile, CaseOrg1 and CaseOrg2 in that they are not driven by commercial business incentives.

5.2 Research Goal 2

With the Community Strategy Framework (CSF), *Paper III* also introduces the concept of community strategy that describes if an organization is in need of such influence in a certain community, and how they should adapt their community engagement to gain it. The importance of having an influence on an OSS community's RE process was first exemplified in *Paper I*. By building and maintaining a symbiotic relationship with communities of the Jenkins and Gerrit OSS projects, Sony Mobile was able to influence the development of the two communities. The importance of influence was further highlighted in *Paper II* as a necessity in order to execute on contribution strategies and to maintain control of important software artifacts once they are contributed.

A community strategy may hence both provide input to and help to execute a contribution strategy. This relationship is further visualized when comparing the

CSF presented in *Paper III* and the CoMn framework in *Paper V*. The CSF presents a number of aspects that may be used to determine the need for influence, while the CoMn framework presents a number of contribution complexities where influence may be needed to manage related risks. For example, one of the complexities in the CoMn framework describes the risks of losing control of the development or that misalignment between the internal and external agendas may arise. Another complexity describes the risk of having too low influence from the start which could complicate a contribution.

By identifying OSS projects and related communities that are important, an organization may work proactively to enable future contributions, but also influence the direction of the development in general. This may be a decisive factor for whether the organization will be able to gain the benefits it expects from the community. As pointed out both in literature [131, 138, 159, 194], as well as in *Papers I-V*, an organization engaged in an OSS community is a stakeholder among many which may introduce conflicting agendas, result in a lack of control over what requirements that are implemented, and miss-alignment with the organization's internal RE process [39, 231].

The engagement practices proposed in the CSF describe ways in how an organization may achieve the aforementioned influence on the RE process within OSS communities with meritocratic coordination processes [198] present. *Paper I's* report on how Sony Mobile's Tools department gained there influence in the Jenkins and Gerrit OSS communities align with the engagement practices proposed by the CSF. In *Paper III*, this is further exemplified in a fictitious example of how the CSF could be applied based on the case study presented in *Paper I*.

5.3 Research Goal 3

In regards to *RG3*, the Stakeholder Influence Analysis (SIA) method (*Paper IV*) can help an organization to both identify and analyze the influence of stakeholders in an OSS community.

Stakeholder populations in OSS communities can be characterized as constantly evolving with new and unknown stakeholders [131]. There are no roosters of members present due to the informal and decentralized characteristics of OSS communities. The case study of the Apache Hadoop OSS community [132], which *Paper IV* extends, shows how the Apache Hadoop community has an evolving population where organizations' influence and collaboration fluctuates with time.

This highlights the need to stay aware of a community's evolving and dynamic stakeholder population as the fluctuating population may otherwise unknowingly introduce conflicting agendas. Depending on the focal organization's position and role in the community's governance [10] and network structure [189], it may be that the focal organization is no longer the vantage point in the OSS community. Hence, organizations may need to analyze the influence of other stakeholders in

order to respond to potential threats towards their own agenda and competitive edge [62].

Papers III and *V* further point out that such analysis may provide input to both contribution and community strategies. Regarding contribution strategies, the presence of (potential) competitors may, e.g., affect the risk of giving away competitive edge, while the presence of organizations with certain expertise may provide an opportunity to extract valuable knowledge. For community strategies, the presence of a (potential) competitor or partner does not have to imply that a certain level of influence on the RE process is warranted. Further information is needed about the agendas of these other stakeholders.

The importance of stakeholder analysis was further exemplified in *Paper I* which describes how Sony Mobile sought out those with similar needs to build traction for the common agendas within the Jenkins and Gerrit communities. Feature-by-feature collaborations were also observed why it may be important to also know who was a common interest even on a specific requirements level. *Paper II* also emphasizes the need to identify those with a similar agenda as they could provide a suitable partner for creating new and smaller communities if e.g., the general interest in a community is limited in regards to a certain feature, or if it is deemed that extra control of the feature is needed.

6 Threats to Validity and Ethical Aspects

In this section, threats to validity and ethical aspects in regards to *Paper I-V* are discussed, as well as how these were managed.

The iterative approach implied by the design science research approach has helped to both validate and refine the artifacts presented in papers included in this thesis. In *Paper III* for example, an iterative interview survey was used to design and refine the CSF until a point of saturation was reached. *Papers II* and *V* presents three design cycles resulting in the CAP model after the first cycle, and the CoMn framework after the second and third cycle. By continuously refining the artifacts through new iterations, a deeper understanding is gained about the problem context while also understanding and improving the internal and external validity [191] of the artifact. However, the artifacts have only been validated in a modeled version of the problem context and have yet to be implemented and evaluated in a real-world problem context [79, 228]. This form of validation may further be viewed as pilot testing which is a recommended practice before solutions are introduced for use in industry [69].

Descriptive validation through the use of scenarios [90] has been used in *Papers II, III* and *IV* to demonstrate the different artifacts' utility and applicability. These scenarios provide proof-of-concept of the artifacts and helped to surface and identify potential issues early on, helping to refine the artifacts further. The scenarios further help to communicate how the artifacts may be applied as well as

information about the problem contexts they are intended for. To further help communicate such information, quotes from interviewees have been used extensively to describe the artifacts in *Paper III* and *V*. This can potentially help to provide further contextual factors that may otherwise risk being lost in the reporting of the research if abstracted by the researcher.

Qualitative data from interviews with practitioners have provided an important foundation for the research results presented in *Papers I, II, III* and *V*. This has been a deliberate design decision made in order to gain in-depth knowledge about the problem context and maintain the relevance of the research. By consulting experts, research can exploit their experience and knowledge about contextual factors to both design and validate the artifacts adapted to the modeled version of the problem context [228]. Semi-structured interviews have predominately been used with questionnaires that have been reviewed between co-authors. Interviews have been audio-recorded and transcribed. Member-checking and continuous feedback-loops were used in all four studies to improve the construct validity [191].

A collaborative research effort was used to maintain the reliability of the research and reducing the risk of researcher bias [191]. In *Papers I-II* all research design, data collection and analysis was performed iteratively among the co-authors. In *Paper III-V* the author of this thesis was responsible for the design, execution, analysis, and reporting. Peer-debriefing through design-inputs and reviews were however provided iteratively during the research process from the co-authors.

In regards to the data collection and analysis in *Papers III* and *V*, the risk for researcher bias is however still present as these steps were performed the first author only. In *Paper III*, this was mitigated through the iterative interview process in the validation phase where interviewees from the problem investigation phase were revisited, along with new experts who had not been interviewed earlier. Member-checking was also used in *Paper V* where interview summaries were presented to key interviewees after the interviews had been transcribed. The same key interviewees were also presented with the framework and asked questions about whether something was redundant, missing, or could potentially be modified.

Compared to *Papers II, III* and *V*, *Paper IV* is limited to presenting the solution artifact together with a proof-of-concept demonstration through the case study of Apache Hadoop OSS community. Further validation is needed to make further claims in regards to SIA's validity.

Regarding the external validity of the papers included in this thesis, one has to consider the problem context which they report. For example in *Papers I, II* and *V*, the way in how the case organizations use OSS may be an important contextual factor to consider. As reported, all organizations use OSS as part of their tools and infrastructure setups, while other use cases do apply for the different organizations. Another perspective is given by CaseOrg3 in *Paper V* which is a public sector organization in contrast to the other case organizations studied. Hence, the cases do provide extremes [59] to each other while still having resembling characteristics.

The limitations of case studies are acknowledged and this thesis does not claim

any statistical generalization [191] in this regard. However, case studies do provide a mean to gather deep knowledge of industry practice and rationale in the problem context [228]. By considering the case organizations' characteristics, the reader can put the CAP model and the CoMn framework as well as the organizations' rationale and concerns for sharing software as OSS into context. Through analytical generalization (cf. analogical inference [228]), results from this study can then be extended to cases with similar characteristics within a similar context [191]. Both similarities and dissimilarities between the source and target cases should be thoroughly analyzed [76].

In regards to *ethical aspects*, careful precautions were taken not to reveal sensitive data from the case organizations 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 organization 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 *Papers I, II* and *V*, non-disclosure agreements were used, as well as continuous feedback-loops with case organization representatives in order to review the level of sensitivity of what is reported, while maintaining the researchers' integrity and independence. *Paper IV* is not tied to any specific case organizations and all data used is available publicly due to the nature of OSS.

7 Future Work

The design science research approach used in this thesis has been limited to the design cycle and focused on designing and validating the different artifacts in a modeled version of the problem context [228]. Future research should, therefore, look to both continuing with further design cycles and when deemed appropriate initiate a technology transfer to a real-world context where the artifact may be implemented and evaluated [228].

For *RG1*, future design cycles should focus on using empirical research methods gathering quantitative data. *Papers I, II* and *V* has focused on case studies which is a good method for gathering deep knowledge of industry practice and investigating the problem context [228]. A survey should aim to generalize the identified contribution objectives and complexities both within and beyond the current problem context which is defined by the case organizations already studied. The population should be limited to organizations that are mature in the way they leverage OSS [157], where the presence of an Open Source Program Office may be one potential indicator [151]. The questionnaire needs to be carefully designed to capture demographics and contextual factors so that the problem context can be defined as clearly as possible.

For *RG2*, future design cycles should investigate the CSF in a real-world context to provide further evidence to its validity, preferably through a multiple-case study [3]. The example presented in *Paper III* which is based on the case study from *Paper I* does provide some guidance for practitioners but further qualitative insights are needed.

For *RG3*, future design cycles could either have a qualitative or quantitative focus. By applying SIA within a case organization, further problem understanding may be gained from the perspective of the practitioners and intended users of the method, specifically in regards to the more qualitative step of applying the influence/agenda alignment matrix. Another potential study could focus on validating the centrality measures and how influence is interpreted by applying the SIA on a number of OSS communities. Samples of the respective communities could then be asked to verify the results and interpretations either through a survey or a series of interviews.

Beyond the continued research in regards to *RG1-3*, this thesis also proposes two new research goals. The first of these concerns the importance of health and sustainability of OSS communities [214], which is emphasized both in *Papers III* and *V*. If an organization is to gain any benefits as defined by the contribution objectives, concerned communities need to have an active stakeholder population that engages and contributes in the many ways as proposed by the CSF. Future research should hence identify challenges for maintainers to OSS projects and specific guidelines for an organization that indicates how and when they can contribute, if *"...not for influence, but for health"* as stated by I1 in *Paper III*. Health in this regard should be specifically investigated and defined considering both the opinions of OSS maintainers, as well as research[21] and industry practice[22]. A fourth research goal for future research is therefore:

RG4: To design a solution that supports software-intensive organizations identify when and how they may contribute to improving an Open Source Software community's health.

The second and last new research goal proposed follows from *Paper V* which shows that software-intensive organizations using OSS can also be found within the public sector. As reported, CaseOrg3 differs from Sony Mobile, CaseOrg1 and CaseOrg2 in that they are not driven by commercial incentives. Instead, they wish to help private actors focus on more value-adding activities and thereby improve job-matching capabilities for employers and job-seekers. This difference in incentives and its potential impact on best practices should be further investigated, as well as the interplay OSS and open data which was also identified as a topic of relevance for CaseOrg3. A fifth research goal for future research is therefore:

[21] See, e.g., https://soheal.github.io/
[22] See, e.g., https://chaoss.community/

RG5: To identify best practices for public sector software-intensive organizations in how they can make use of OSS and open data to improve the services they provide.

8 Conclusions and Main Contributions

The objective of this thesis is two-fold. The first objective is to create guidance for organizations in making decisions of what to share as OSS that are in line with the organization's business goals. The second objective is to create guidance for how an organization can identify OSS communities where they need to have an influence on the RE process, and how they can gain it, in order to achieve its internal agenda.

To address these two objectives, the two concepts of contribution and community strategies are introduced. Contribution strategies answer the questions if a software artifact (e.g., a feature or project) or parts of it should be released as OSS, when in time, and if it should be contributed to an existing OSS community, or if a new community should be established. Community strategies answer the questions what OSS communities an organization views as important and need to have an influence on in terms of their RE process, and also how this influence may be gained.

A design science research approach is used to investigate and understand the underlying problems in their contexts and develop knowledge that can be used to design artifacts that may provide an improvement, or solve the problems in question in the real world. Three research goals are therefore defined, representing design problems which *"calls for a change in the world"* [228]. The research goals are addressed through *Papers I-V* by answering a number of research questions, representing knowledge questions which call for knowledge about the world as is [227]. Using this distinction, the main contributions of this thesis can be considered as two-fold. In part by the artifacts that may provide an improvement, or solve the problems in question in the real world, and in part by the design knowledge captured in these artifacts that may help to better understand the problem contexts and how the artifacts may be adapted to their contextual factors.

Following the recommendations by Storey et al. [204], the main contributions of this thesis are summarized in the following Technological Rules (TR), related to the three research goals:

TR1: To achieve alignment between business goals and what is shared as OSS, software-intensive organizations should develop contribution strategies as well as contribution strategy guidelines and related contribution processes for internally developed software artifacts using the CAP model and the CoMn framework presented in *Papers II* and *V* respectively.

TR2: To identify OSS communities that align with business goals and achieve its internal agenda within these communities, software-intensive organizations should develop community strategies using the CSF presented in *Paper III*.

TR3: To create a contextual awareness of an OSS community's stakeholder population and input its contribution and community strategies, software-intensive organizations should analyze stakeholders' influence on the community's RE process using the SIA method presented in *Paper IV*.

INCLUDED PAPERS

CHAPTER I

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

Hussan Munir, Johan Linåker, Krzysztof Wnuk, Per Runeson and 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 are not restricted to product innovation but also include process innovation, e.g. improved SE practices and methods. **Aim.** This study explores the involvement of a software organization (Sony Mobile) in OSS communities from an OI perspective and what SE practices (requirements engineering and testing) have been adapted in relation to OI. It also highlights the innovative outcomes resulting from OI. **Method.** An exploratory embedded case study investigates how Sony Mobile use and contribute to Jenkins and Gerrit; the two central OSS tools in their continuous integration tool chain. Quantitative analysis was performed on change log data from source code repositories in order to identify the top contributors and triangulated with the results from five semi-structured interviews to explore the nature of the commits. **Results.** The findings of the case study include five major themes: i) The process of opening up towards the tool communities correlates in time with a general adoption of OSS in the organization. ii) Assets not seen as competitive advantage

nor a source of revenue are made open to OSS communities, and gradually, the organization turns more open. iii) The requirements engineering process towards the community is informal and based on engagement. iv) The need for systematic and automated testing is still in its infancy, but the needs are identified. v) The innovation outcomes included free features and maintenance, and were believed to increase speed and quality in development. **Conclusion.** Adopting OI was a result of a paradigm shift of moving from Windows to Linux. This shift enabled Sony Mobile to utilize the Jenkins and Gerrit communities to make their internal development process better for its software developers and testers.OI was a result of a paradigm shift of moving from Windows to Linux. This shift enabled Sony Mobile to utilize the Jenkins and Gerrit communities to make their internal development process better for its software developers and testers.

1 Introduction

Software organizations have recently been exposed to new facets of openness that go beyond their experience and provide opportunities outside their organizational walls. Chesbrough [30] explains the term *Open Innovation* (OI) as *"a paradigm that assumes that organizations can and should use external ideas as well as internal ideas, and internal and external paths to market, as they look to advance their technology"*. OI is based on *outside-in* and *inside-out* knowledge flows that help to advance technology and spark innovation. Some classical examples of inside-out are selling intellectual property while outside-in correspond to start-up acquisition and integration. There are also *coupled processes* [53] where companies give and take during co-creation by making alliances and joint-ventures. OI is fuelled by increased mobility of workers and knowledge, more capable universities, greater knowledge access and sharing capabilities that World Wide Web offers [34] and easier access to venture capital for start-ups.

Open Source Software (OSS) was widely used by software organizations before the OI model became popular [123] and nowadays provides a common example of OI [159]. OSS leverages external resources and knowledge to increase innovation, product quality and to shorter time-to-market. OSS offers not only potential product innovation (e.g. by using an OSS platform of commodity parts to build differentiation parts), but potential process innovations in terms of an implementation of new or significantly improved production or delivery methods [127].

IBM's engagement in the Linux community in terms of patent and monetary contributions exemplifies how a firm can leverage OSS from an OI perspective. Risks and costs of development were in this case shared among other stakeholders such as Intel, Nokia, and Hitachi, which also have made significant investments in the Linux community [122]. Thanks to Linux involvement, IBM can strengthen its own business model in selling proprietary solutions for its clients running on

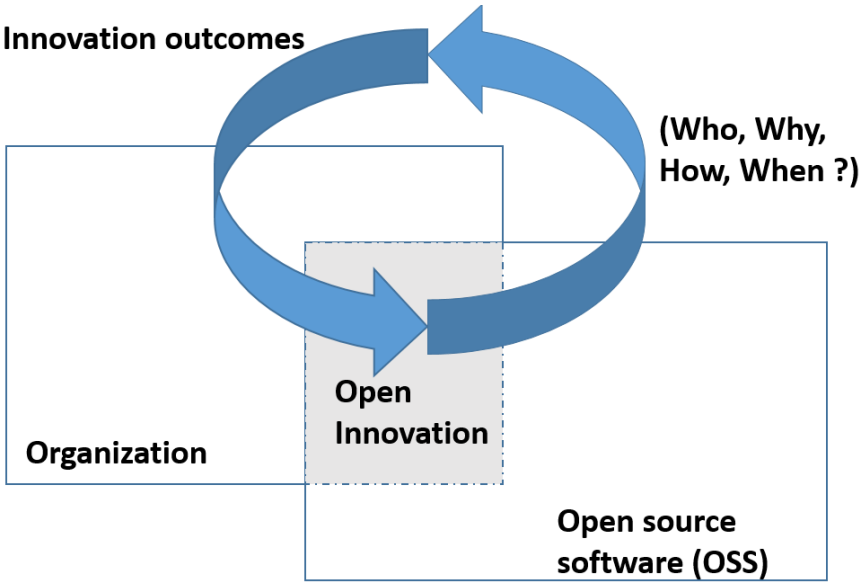


Figure 1: Study Objectives in the intersection between proprietary organizations and open source software.

top of Linux. Additionally, the openness of Linux also gave IBM more freedom to co-develop products with its customers [34].

Software organizations that want to benefit from OI via OSS engagement need to adapt and innovate their internal software development strategies and processes. For example, influence on feature selection and road-mapping may be gained through a more active participation, as many OSS communities are based on meritocracy principles [101]. Also, some benefits may first be fully utilized after contributing back certain parts to the OSS community [213]. For example, by correcting bugs, actively participating in discussions and contributing new features, a software organization might reduce maintenance cost compared to proprietary software development [205]. Hence, in order for a firm to gain the expected benefits of products, OI process innovations may be a required step on the way forward [119, 188, 231]. Existing literature does not particularly focus on how these internal SE process adaptions should be structured or executed [159]. Further, little is known about how OSS involvement may be utilized as an enabler and support for further innovation spread inside an organization, e.g. process, tools, or organizational innovations.

In this study, we focus on identifying when, why and how a software organization adopts OI through the use of OSS, and what innovative outcomes can be gained (see Fig. 1). We investigate these aspects through a case study at Sony

Mobile and how it actively participate and contribute to the communities of the two OSS tools Jenkins and Gerrit. These two tools are the basis of Sony Mobile's internal continuous integration tool chain. The study further investigates how external knowledge and innovation captured through the active development of these OSS tools may be transferred into the product development teams of Sony Mobile. More explicitly, this study contributes by studying how OSS may be used, not only for leveraging product innovation [127] in the tools themselves, but also how these tools can be used as enablers for process innovation in the form of improved SE practices and product quality.

This paper is structured as follows. Section 2 highlights the related work and Section 3 outlines the research methodology. In Sections 4 and 5 results from the quantitative and qualitative analysis are presented, respectively. Finally, Section 6 discussed the results, followed by conclusions in Section 7.

2 Related work

In this section, we summarize related work in OI strategies, OI challenges in SE and open source development practices inside software organizations. This section is partly based on the systematic mapping study by Munir et al. [159].

The increased openness that OI implies poses significant challenges to software organizations in terms of securing their competitive advantage [159] and understanding what to contribute, when and how to maintain differentiation towards competitors that may also be involved in the OSS community [86, 98, 212]. Related to that is the challenge of what requirements should be selected, when these should be released and how an internal roadmap should be synchronized with the OSS project's roadmap [132, 231]. These challenges highlight the need for a clear contribution strategy that software organizations should create to focus their internal resources on value-creating activities, rather than contributing unnecessary patches or differentiating features [231].

Extensive involvement in OSS communities may also bring significant challanges. Among these challenges, Daniel et al. [44] suggested that the conflict between organizational and OSS standards reduces developers' organizational commitment and it is strongly dependent on the degree to which developers associate themselves with organizations or OSS communities. Investing in OSS may also be costly and create differentiation and property right protection challenges, as indicated by Stuermer et al. [205] who studied the Nokia Internet Tablet, which was based on a hybrid of OSS and proprietary software development.

West et al. [225] examined the complex ecosystem surrounding Symbian Ltd. and identified three inherent difficulties for organizations leading an OI ecosystem: 1) prioritizing the conflicting needs of heterogeneous ecosystem participants, 2) knowing the ecosystem requirements for a product that has yet to be created, and 3) balancing the interests of those participants against those of the ecosystem leader.

Looking at OI strategies, Dahlander & Magnusson [39] show how organizations may access OSS communities in order to extend the firm's resource base, align the organization's strategy with that of the OSS community, and/or assimilate the community in order to integrate and share results with them. The same authors explained that depending on how open a firm chooses to be in regards to their business model, different strategies may be enforced, e.g. symbiotically giving back result to the community, or as a free-rider keeping modifications and new functionality to oneself [38]. Some strategies include:

- selectively revealing - differentiating parts are kept internal while commodity parts are made open [86, 219]. This requires continuous assessment of what parts are to be considered commodity as opposed to differentiating value.
- licensing schemas (cf. Dual-licensing [32]), technology may be fully disclosed, but under a restrictive license [219]. Alternatively, everything may be disclosed under open and transparent conditions [32].

Henkel [86] reports how small organizations reveal more, as they are likely to benefit from the external development support. Component manufacturers also reported to contribute a lot as they have a good protection of the hardware they sell; software is seen as a complementary asset. In a follow-up study, Henkel [88] further reported how openness had become a competitive edge, as customers had started to request even more revealing.

Dahlander & Wallin [40] show how having an employee in the community can be an enabler for the organizations to not only gain a good reputation but also to influence the direction of the development towards the organizations' own interests. However, to gain the roles needed to commit or review code written by community developers, individuals need to contribute and become an active part of the communities as these are often based on the principles of meritocracy [101].

Inner Source [203] has gained interest among researchers and practitioners as a way to adapt OSS practices at software organizations. Such hybrids of commercial and OSS practices [148] could include using the OSS style project structure, where a core team of recognized experts has the power to commit code to an official release, and a much larger group contributes voluntarily in many ways.

Summary
Research has shown a lot of interest for OI and its different applications [221], including leveraging OSS for OI [159]. However, the focus is mostly limited to management and strategic aspects, e.g., [39, 205, 226], with some exception of inner sourcing [152, 203]. Little is still known about what triggers software organizations to adopt OSS from an OI perspective and how this affects SE practices [159].

Table 1: Research questions with description

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

This paper adds to existing knowledge by focusing on the use of OSS from an OI perspective in an organization that seek to complement its internal product development and process innovation [127] with the use of external knowledge from OSS communities. Furthermore, this study aims to improve our understanding of what and how a software organization can open up and how SE practices are adapted to deal with the openness to OSS communities.

3 Case study design

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

3.1 Research questions

The focus of this study is on how software organizations use OSS projects from an OI perspective, what triggers them to open up and how this impacts the organizations' 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 [169] and Gerrit [94]. Both tools constitute pivotal parts in Sony Mobile's internal continuous integration tool chain.

The study further investigates how external knowledge and innovation captured through the development of these OSS tools, may be transferred into the product development teams of Sony Mobile. More explicitly, this study contributes by studying how OSS may be used, not only for leveraging product innovation [127] in the tools themselves, but also how these tools can be used as enablers for process innovation in the form of improved SE practices and tools within the organization.

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

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

Based on this background, and the research gap identified in earlier work [159], we formulate our research questions to study the OI in Sony Mobile in an exploratory manner (see Table 1). *RQ1* addresses the extent to which Sony Mobile is involved in the Jenkins and Gerrit communities and its key contribution areas (i.e. bug fixes, new features, documentation etc.). *RQ2* and *RQ3* explore the rationale behind Sony Mobile's transition from closed innovation to OI. *RQ4* highlights the key innovation outcomes realized as a result of openness. Finally, *RQ5* aims at understanding whether or not the existing requirements engineering and testing processes have the capacity to deal with the OI challenges in SE. *RQ1* is answered with the help of quantitative analysis of repository data, while the remaining four research questions (*RQ2, RQ3, RQ4, RQ5*) are investigated using qualitative analysis of interview data.

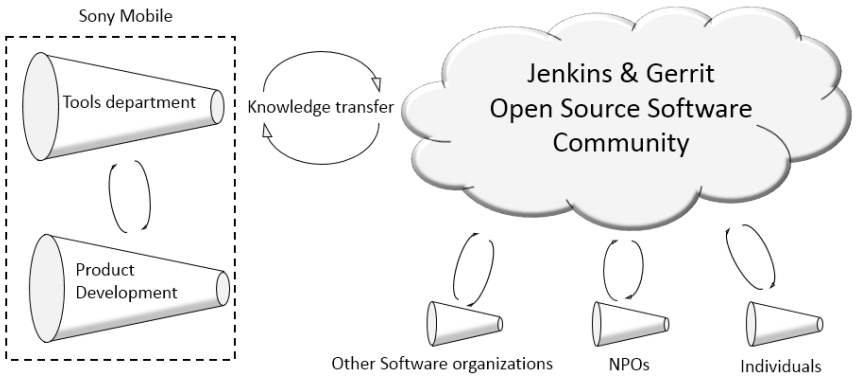


Figure 2: The Jenkins and Gerrit OSS communities surrounded by Sony Mobile and other members. Arrows represent knowledge transfer in and out of the community members such as other software organizations, non profit organizations (NPO) and individuals, which in turn are illustrated by funnels, commonly used in OI literature [30].

3.2 Case Selection and Units of Analysis

Sony Mobile is a multinational corporation with roughly 5,000 employees, developing embedded devices. The studied branch focuses on developing Android-based phones and tablets and has 1600 employees, of which 900 are directly involved in software development. Sony Mobile develops software in an agile fashion and applies software product line management with a database of more than 20,000 features suggested or implemented across all product lines [180].

However, in order to work with OSS communities, namely Jenkins and Gerrit Sony Mobile created a designated tools department to acquire and integrate the external knowledge to improve the internal continuous integration process. The continuous integration tool chain used by Sony Mobile is developed, maintained and supported by an internal tools department. The teams working on phones and tablets are thereby relieved of this technical overhead. During the recent years, Sony Mobile has transitioned from passive usage of the Android codebase into active involvement and community contribution with many code commits to Jenkins and Gerrit. This maturity resulted in a transition from closed innovation to OI [30], assuming that business values are created or captured as an effect.

From an OI perspective, there are interactions between the Tools department and the Jenkins and Gerrit communities (see Fig. 2). The in- and outgoing transactions, visualized by the arrows in Fig. 2, are data and information flows, e.g. ideas, support and commits, can be termed as a coupled innovation process [53]. The exchange is continuous and bi-directional, and brings product innovation into the Tools department in the form of new features and bug fixes to Jenkins and Gerrit.

The Tools department can, in turn, be seen as a gate between external knowledge and the other parts of Sony Mobile (see Fig. 2). The Tools department accesses, adapts and integrates the externally obtained knowledge from the Jenkins and Gerrit communities into the product development teams of Sony Mobile. This creates additional transactions inside Sony Mobile which can be labeled as process innovation [1] in the sense that new tools and ways of working improve development efficiency and quality. This relates to the internal complementary assets need that is mentioned as an area for future research by Chesbrough et al. [33].

We conducted a case study design with Jenkins and Gerrit as units of analysis [191] as these are the products in which the exchange of data and information enable further innovation inside Sony Mobile.

3.3 Case study procedure

We performed the following steps.

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

3.4 Methods for quantitative analysis

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

Preliminary Investigation of Jenkins and Gerrit Commits

A *commit* is a snapshot of a developer's files after reaching a code base state. The number of lines of code in a commit may vary depending upon the nature of the commit (e.g. new implementation, update etc.) [83]. The comment of a commit refers to a textual message related to the activity that generates the updated new piece of code. It ranges from a simple note to a detailed description, depending on the project's conventions. In this study, we used the keywords provided by Hattori and Lanza [83] in his study as a reference point to classify the commit messages (see Table 2).

We mined the source code repositories of Jenkins and Gerrit to extract the commit id, date, committer name, committer email and commit description message for each commit, with the help of the tool CVSAnlY [145]. The extracted data was stored locally in a relational database with a standard data scheme, independent of the analyzed code repository. The structure of the database allows a quantitative analysis to be done by writing SQL queries. The number of commits per committer were added together with the name and email of the committer as keys.

We extracted the affiliations of the committers from their email addresses by filtering them on the domain, e.g., john.doe@sonymobile.com was classified with a Sony Mobile affiliation. It is recognized that committers may not use their corporate email addresses when contributing their work, since parts of their work could be contributed privately or under the umbrella of other organizations than their employer. To triangulate and complement this approach, a number of additional sources were used, as suggested by earlier research [19, 78]. First, social media sites as LinkedIn, Twitter and Facebook were queried with keywords from the committer, such as the name, variations of the username and e-mail domain. Second, unstructured sources such as blogs, community communication (e.g., comment-history, mailing-lists, IRC logs), web articles and firm websites were consulted.

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

- Gerrit[1]
- PyGerrit (Gerrit plug-in)[2]
- Gerrit-events (Gerrit plug-in)[3]
- Gerrit-trigger (Jenkins plug-in)[4]
- Build-failure-analyzer (Jenkins plug-in)[5]
- External-resource-viewer (Jenkins plug-in)[6]
- 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 and Lanza [83]. This was done manually by analyzing the description messages of the commits and searching for keywords (see Table 2), and then classifying the commits in one of the following categories:

Forward engineering activities refer to the incorporation of new features and implementation of new requirements including the writing new test cases to verify the requirements. **Re-engineering** activities deal with re-factoring, redesign and other actions to enhance the quality of the code without adding new features. **Corrective engineering** activities refer to fixing defects in the software. **Management activities** are related to code formatting, configuration management, cleaning up code and updating the documentation of the project.

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

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

Table 2: Keywords used to classify commits taken from Hattori and Lanza [83].

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		

3.5 Methods for qualitative analysis

The quantitative analysis had laid a foundation to understand the relation between Sony Mobile, and the Jenkins and Gerrit communities. Therefore, in the next step we added a qualitative view by interviewing relevant people inside Sony Mobile in order to address *RQ2–RQ5*. Interview questions are listed in the Appendix.

Interviewee selection

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

Table 3: Interviewee demographics.

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

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 organization dedicated to Jenkins development. His responsibilities as the tools manager for Jenkins at Sony Mobile were taken over by interviewee I4. Interviewee I4 is a Software Architect in the Tools department involved further down in Sony Mobile's continuous integration tool chain and gives an alternative perspective on the OSS involvement of the Tools department as well as a higher, more architectural view on the tools. Interviewee I5 is an upper-level manager responsible for Sony Mobile's overall OSS strategy, which could contribute with a top-down perspective to the qualitative analysis.

The interviews were semi-structured, meaning that interview questions were developed in advance and used as a frame for the interviews, but still allowing the interviewers to explore other relevant findings during the interview wherever needed. The two first authors were present during all five interviews, with the addition of the third author during the first and fifth ones. Each interviewer took turns asking questions, whilst the others observed and took notes.

Each interview was recorded and transcribed. A summary was also compiled and sent back to the interviewees for a review. Any misunderstandings or corrections could then be sorted out. The duration of the interviews varied from 45 to 50 minutes.

3.6 Validity threats

This section highlights the validity threats related to the case study. Four types of validity threats [191] are addressed with their mitigation strategies.

Internal validity

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

Confounding factors. To mitigate the risk of introducing confounding factors, the study was performed on the tools level instead of an organizational level to ensure that the innovation outcomes are merely the result of adopting OI. Performing the study on an organization level introduces the risk of confounding the innovation outcomes as a result of a product promotion or financial investment etc. instead of the use of external knowledge from OSS communities. Therefore, a more fine-grained analysis on the OSS tools level was chosen to minimize the threat of introducing confounding factors.

Subjectivity. It was found in the study that Sony Mobile does not use any general innovation metrics to measure the impact of OI. Therefore, researchers had to rely on qualitative data. This leads to the risk of introducing subjectivity while inferring innovation outcomes as a result of OI adoption. In order to minimize this risk, the first two authors independently performed the analysis and the remaining authors reviewed it to make the synthesis more objective. Moreover, findings were sent back to interviewees for validation. Furthermore, subjectivity was minimized by applying the commit messages classification criteria proposed by Hattori and Lanza. [83]. During the analysis, the disagreements were identified using Kappa analysis and resolved to achieve a substantial agreement.

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

External validity

This refers to the extent it is possible to generalize the study findings to other contexts. The scope of this study is limited to a software organization utilizing the notion of OI to accelerate its innovation process. The selected case organization is a large-scale organization with an intense focus on software development

for embedded devices. Moreover, Sony Mobile is a direct competitor of all the main stream organizations making Android phones. The involvements by other stakeholders in the units of analysis (Jenkins and Gerrit) indicate their adoption of Google's tool chain to improve their continuous integration process. Therefore, the findings of this study may be generalized to major stakeholders identified for their commits to Jenkins and Gerrit, and other OSS tools used in the tool chain development. Our findings may also be relevant to software organizations with similar context, domain and size as Sony Mobile.

Construct validity

This refers to what extent the operational measures that are studied really represent what researcher has in mind, and what is investigated according to the research questions [191]. We took the following actions to minimize construct validity threats.

Selection of interviewees. We conducted a preliminary quantitative analysis of the Jenkins and Gerrit repositories to identify the top committers and to select the relevant interviewees. The selection was performed based on the individuals' commits to Jenkins or Gerrit. Moreover, recommendations were taken from interviewees for suitable further candidates to attain the required information on OI. Process knowledge, role, and visible presence in the community were the key selection factors.

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

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

Reliability

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

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

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.

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

4 Quantitative analysis

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

4.1 Gerrit

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

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

Table 5: Classification of Sony Mobile's commits to OSS tools based on the criteria by Hattori and Lanza [83]

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

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

PyGerrit

PyGerrit is a Python library that provides a way for clients to interact with Gerrit. As can be seen in Table 6, Sony Mobile initiated this plug-in and is the biggest committer to it, representing 97.5% of the commits. Management commits are the most frequent category, followed by forward engineering commits. This suggests that some code formatting changes, cleaning up code and documentation commits were delivered by Sony Mobile after opening up this plug-in to the community.

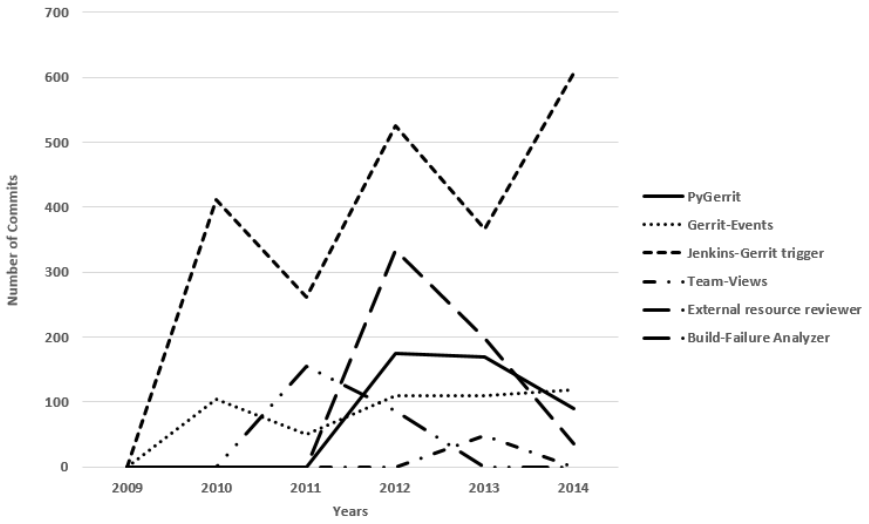


Figure 3: Sony Mobile's commits for all OSS tools per year

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

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

Sony Mobile's yearly contribution analysis shows a steady growth since its introduction in 2011 (see Fig. 3).

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

Gerrit-event

Gerrit-event is a Java library used primarily to listen to stream-events from Gerrit Code Review and to send reviews via the SSH CLI or the REST API. It was originally a module in the Jenkins Gerrit-trigger plug-in and is now broken out to be used in other tools without the dependency to Jenkins. Table 6 shows that apart from Sony Mobile(66.1%), HP(4.1%), SAP(0.2%), Ericsson(3.3%) and Intel(2%) commits reveal that they are also using Gerrit-event in their continuous integration process. Sony Mobile started contributing to Gerrit-event in 2009 and since then seem to be the largest committer along with its competitors (see Table 6). Similarly, to the PyGerrit plug-in, management and forward engineering commits dominate and Sony Mobile is the main driver of features to this community.

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

4.2 Jenkins

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

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

Gerrit-trigger

This plug-in triggers builds on events from the Gerrit code review system by retrieving events from the Gerrit command stream-events, so the trigger is pushed from Gerrit instead of pulled as scm-triggers usually are. Multiple builds can be triggered by one change-event, and one consolidated report is sent back to Gerrit.

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

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

Build-failure-analyzer

This plug-in scans build logs and other files in the workspace for recognized patterns of known causes to build failures and displays them on the build page for quicker recognition of why the build failed. As can be seen in see Table 6, Sony Mobile came out as the largest committer (85.5%) to the Build-failure-analyzer. One possible explanation for the lack of contribution from the other software organizations is that this plug-in might be very specific to the needs of Sony Mobile, but they made it open to see if the community shows interest in contributing to further development efforts.

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

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

External-resource-viewer

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

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

Team-views

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

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

5 Qualitative analysis

We conducted thematic analysis [35, 36] to find recurring patterns in the collected qualitative data. The following steps were performed in the process.

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

5.1 Opening up

The process of opening up for external collaboration and maturing as an open source organization, can be compared to moving from a closed innovation model to an OI model [30]. 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 [226]. Switching to Android correlates to a general shift in the development environment, moving from Windows to Linux. This concerned the

Table 7: Themes emerging from the thematic analysis.

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

tools used in the product development as well. A transition was made from existing proprietary solutions, e.g. the build-server Electric commander, to the tools used by Google in their Android development, e.g. GIT and Gerrit. As stated by I2, *"... suddenly we were almost running pretty much everything, at least anything that touches our phone development, we were running on Linux and open source"*. This was not a conscious decision from management but rather something that grew bottom-up from the engineers. The engineers further felt the need for easing off the old and complex chain of integration and building process.

At the same time, a conscious decision was made regarding to what extent Sony Mobile should invest in the open source tool chain. As stated by I5, *"... not only should [the tool chain] be based on OSS, but we should behave like an active committer in the ways we can control, understand and even steer it up to the way we want to have it"*. The biggest hurdle concerned the notion of giving away internally developed intellectual property rights, which could represent competitive advantage. The legal department needed some time to understanding the benefits and license aspects, which caused the initial contribution process to be extra troublesome. In this case, Sony Mobile benefited from having an internal champion and OSS evangelist (I5). He helped to drive the initiative from the management side, explained the issues and clarified concerns from different functions and levels inside Sony Mobile. Another success factor was the creation of an OSS review board, which included different stakeholders such as legal department representatives, User Experience (UX) design, product development and product owners. This allowed for management, legal, and technology representatives to meet and discuss OSS related issues. The OSS contribution process now includes submitting a form for review, which promotes it further after successful initial screening. Next, the OSS review board gives it a go or no-go decision. As this

would prove bureaucratic if it would be needed for each and every contribution to an OSS community, frame-agreements are created for open source projects with a high-intensity involvement, e.g. Jenkins and Gerrit. This creates a simplified and more sustainable process allowing for a day to day interaction between developers in the Tools department and the communities surrounding Jenkins and Gerrit. Sony Mobile's involvement in OSS communities is in-line with the findings of governance in OSS communities by Jensen [102].

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

5.2 Determinants of openness

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

Another strategic factor concerns having a first-mover advantage. Contributing a new feature or a project first means that Sony Mobile as the maintainer gets a higher influence and a greater possibility to steer it in their own strategic interest. If a competitor or the community publishes the project, Sony Mobile may have less influence and will have to adapt to the governance and requirements from the others. A good example here is the Gerrit-trigger. The functionality was requested internally at Sony Mobile and therefore undergone development by the Tools department during the same period it became known that there was a similar development ongoing in the community. As stated by I3, *"...we saw a big risk of the community going one way and us going a very different route"*. This led to the release of the internal Gerrit-trigger as an open source plug-in to Jenkins, which ended up being the version with gained acceptance in the Jenkins and Gerrit

communities. The initial thought was however to keep it closed according to I3, *"... We saw the Gerrit-trigger plug-in as a differentiating feature meaning that it was something that we shouldn't contribute because it gave us a competitive edge towards our competitors [in regards to our continuous integration process]"*. It should be noted that this was in the beginning of the process of opening up in Sony Mobile and a positive attitude was rising. A quote from I3 explains the positive attitude of the organization which might hint about future directions, *"... in 5 years' time probably everything that Sony Mobile does would become open"*.

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

5.3 Requirements engineering

This theme provides insights about requirements engineering practices in an example OI context. The requirements process in the Tools department towards the Jenkins and Gerrit communities does not seem very rigid, which is a common characteristic for OSS [192]. The product development teams in Sony Mobile are the main customers of the Tools department. The teams are, however, quite silent with the exception of one or two power users. There is an open backlog for internal use inside Sony Mobile where anyone from the product development may post feature requests. However, a majority of the feature requests are submitted via e-mail. The developers in the Tools department started arranging monthly workshops where they invited the power users and the personnel from different functional roles in the product development organization. An open discussion is encouraged allowing for people to express their wishes and issues. An example of an idea sprung out from this forum is the Build-failure-analyzer[8] plug-in. Most of the requirements are, however, elicited internally within the Tools department in a dialogue between managers, architects and developers. They are seen to have the subject matter expertise in regards to the tool functionality. According to I2, there are *"... architect groups which investigate and collaborate with managers about how we could take the tool environment further"*. This is formulated as 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.

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

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

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

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

In the projects where the Tools department is not a maintainer, community governance needs more care. In the Gerrit community, new features are usually discussed via mailing lists. However, large features are managed at hackathons by the Tools department where they can communicate directly with the community to avoid getting stuck in tiny details [152]. As brought up by I2, *"... with the community you need to get people to look at it the same way as you do and get an agreement, otherwise it will be just discussions forever"*. This is extra problematic in the Gerrit community as the inner core team with the merge rights consists of only six people, of which one is from Sony Mobile. One of the key features received from the community was the tagging support for patch sets. I2 stated, *"... When developers upload a change which can have several revisions, it enabled*

us to tag meta-data like what is the issue in our issues handling system and changes in priorities as a result of that change. This tagging feature allows the developers to handle their work flow in a better way". This whole feature was proposed and integrated during a hackathon, and contained more than 40 shared patch sets. Prior to implementing this feature together with the community (I3 quoted) *"... we tried to do it with the help of external consultants but we could not get it in, but meeting core developer in the community did the job for us".*

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

This close collaboration with the community is important as Sony Mobile does not want to end up with an internal fork of any tool. An I2 quoted, *"If we start diverging from the original software we can't really put an issue in their issue tracker because we can't know for sure if it's our fault or their system and we would loose the whole way of getting help from community to fix stuff and collaborate on issues"*. Another risk would be that *"... all of a sudden everybody is dependent on stuff that is taken away from the major version of Gerrit. We cannot afford to re-work everything"*. Due to these reasons, the Tools department is keen on not keeping stuff for themselves, but contributing everything [213, 231]. An issue in Jenkins is that there exist numerous combinations and settings of plug-ins. Therefore, it is very important to have backward compatibility when updating a plug-in and planning new features.

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

5.4 Testing

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

Bugs and issues are, similarly to feature requests, reported internally either via e-mail or an open backlog. Externally, bugs or issues are reported via the issue trackers available in the community platforms. The content of the issue trackers is based on the most current pressing needs in the Tools department. Critical issues are prioritized via the Kanban speed lane which refers to a prioritized list of requirements/bugs based on the urgent needs of Sony Mobile. If a bug or an issue has low priority, it is reported to the community. This self-focused view correlates with the mentality of how the organization would benefit from making a certain contribution, which is described to apply externally as well, *"... Organizations take the issues that affect them the most"*. However, it is important to show to the community that the organization wants to contribute to the project as a whole and not just to its parts, as mentioned by Dahlander [40]. In order to do so, the Tools department continuously stays updated about the current bugs and their status. It is a collaborative work with giving and taking. *"Sometimes, if we have a big issue, someone else may have it too and we can focus on fixing other bugs so we try to forward as many issues as possible"*.

In Gerrit, the Tools department is struggling with an old manual testing framework. Openness has lead them to think about switching from the manual to an automated testing process. I2 stated, *"... It is one of my personal goals this year to figure out how we can structure our Gerrit testing in collaboration with the community. Acceptance tests are introduced greatly in Gerrit too but we need to look into and see how we can integrate our tests with the community so that the whole testing becomes automated"*. In Jenkins, one of the biggest challenges in regards to test is to have a complete coverage as there are many different configurations and setups available due to the open plug-in architecture. However, Gerrit still has some to catch up as stated by I2, *"it is complex to write stable acceptance tests in Gerrit as we are not mature enough compared to Jenkins"*. A further issue is that the test suites are getting bigger and therefore urges the need for automated testing.

Jenkins is considered more mature since the community has an automated test suite which is run every week when a new version of the core is released. This test automation uses Selenium[9], which is an external OSS test framework used to facilitate the automated acceptance tests. It did not get any traction until recently because it was written in Ruby, while the Jenkins community is mainly Java-oriented. This came up after a discussion at a hackathon where the core members in the community gathered, including representatives from the Tools 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

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

tests to the community and have the community execute what Sony Mobile tests when setting up the next stable version. Consequently, it requires less work of Sony Mobile when it is time to test a new stable version. From the community perspective I3 stated, *"an Acceptance Test Harness also helps the community and other Organizations to understand what problems that big or small organizations have in terms of features or in terms other requirements on the system. So it's a tool where everyone helps each other"*.

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

5.5 Innovation outcomes

The word *innovation* has a connotation of newness [7] and can be classified as either things (products and services), or changes in the way we create and deliver products, services and processes. Assink [7] classified innovation into disruptive and incremental. Disruptive innovations change the game by attacking an existing business and offering great opportunities for new profits and growth. Incremental innovations remain within the boundaries of the existing technology, market and technology of an organization. The innovation outcomes found in this study are related to incremental innovations.

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

1. Free features.
2. Free maintenance.
3. Freed-up time.
4. Knowledge retention.
5. Flexibility in implementing new features and fixing bugs.
6. Increased turnaround speed.
7. Increased quality assurance.
8. Improved new product releases and upgrades.
9. Inner source initiative.

The most distinct innovation outcome is the notion of obtaining ***free features*** from the community, which have different facets [39,205]. For projects maintained by Sony Mobile, such as the Gerrit-trigger plug-in, a noticeable amount of external commits can be accounted for. Similarly, in communities where Sony Mobile is not a maintainer, they can still account for free work, but it requires a higher effort in lobbying and actively steering the community in order to maximize the benefits for the organization. Along also comes, the ***free maintenance*** and quality assurance work, which renders better quality in the tools. Furthermore, the use of tools (Jenkins and Gerrit) helped software developers and testers to better manage their work-flow. Consequently, it ***freed-up time*** for the developers and testers that could be used to spent on other innovation activities. The observed innovation example in this case was the developers working with OSS communities, acquiring and integrating the external knowledge into internal product development.

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

An organizational innovation outcome of opening up is the ***knowledge retention*** which comes from having a movable workforce. People in the community may move around geographically, socially and professionally but can still be part of the community and continue to contribute. I3, who took part in the initiation of many projects, recently left Sony Mobile but is still involved in development and reviewing code for his former colleagues which is in line with the findings of previous studies [152,205]. Otherwise, the knowledge tied to I3 would have risked being lost for Sony Mobile.

Sony Mobile had many proprietary tools before opening up. Adapting these tools, such as the build server Electric commander, was cumbersome and it took long time before even a small fix would be implemented and delivered by the supplier. This created a stiffness whereas open source brought ***flexibility***. I2 quoted, *"...Say you just want a small fix, and you can fix that yourself very easily but putting a requirement on another organization, I mean it can take years. Nothing says that they have to do it"*. This increase in the ***turnaround speed*** was besides the absence of license fees, a main argument in the discussions when looking at Jenkins as an alternative to Electric commander. This was despite the required extra involvement and cost of more internal man-hours. As a result, the continuous integration tool chain could be tailored specifically to the needs of the product development team. I1 stated that *"...Jenkins and Gerrit have been set up for testers and developers in a way that they can have their own projects that build code and*

make changes. Developers can handle all those parts by themselves and get to know in less than 3 minutes whether or not their change had introduced any bugs or errors to the system". Ultimately, it provides ***quality assurance*** and performance gains by making the work flow easier for software developers and testers. Prior to the introduction of these tools there was one engineer who was managing the builds for all developers. In the current practice everybody is free to extend on what is given to them from tools department. It offers more scalability and flexibility [153].

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

The obtained external knowledge about the different parts of the continuous integration tool chain enabled better product development. However, the Tools department has to take the responsibility for the whole tool chain and not just its different parts, e.g. Jenkins and Gerrit, described by I5 as the next step in the maturity process. The tool chain has the potential to function as an enabler in other contexts as well, seeing Sony Mobile as a diversified organization with multiple product branches. By opening up in the way that the Tools department has done, effects from the coupled OI processes with Jenkins and Gerrit may spread even further into other product branches, possibly rendering in further innovations on different abstraction levels [127]. 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 [218]. The tool chain is even seen as the foundation for a platform which is supposed to facilitate this sharing [126]. The Tools department is considered more mature in terms of contributing and controlling the OSS communities. Hence, the Tools department can be used as an example of how other parts of the organization could open up and work with OSS communities. I5 uses this when evangelizing and working on further opening up the organization at large, and describes how *"...they've been spearheading the culture of being active or in engaging something with communities"*.

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

6 Results and discussion

Results from the quantitative and qualitative analysis are discussed below, of which the latter is addressed per theme, and connected to the research questions defined in Table 1. Table 8 presents the mapping of research questions to answers with section numbers. Furthermore, a brief summary of answers to research questions is highlighted in section 7.

Table 8: Mapping of answers to RQs with section numbers

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

6.1 Involvement of Sony Mobile in OSS Communities

Addressing **RQ1** in Table 1, the quantitative analysis showed that Sony Mobile has an active role in numerous OSS projects. In most of the analysed projects, Sony Mobile is the initiator and maintainer. An exception is Gerrit where they entered an already established project. However, with 8.2 % (see Table 6) of the commits during the investigated time-span, they have established themselves in the community and been able to contribute the necessary adaptions for Gerrit to function as a part of the continuous integration tool-chain used inside Sony Mobile. This shows that Sony Mobile has an open mindset to creating their own OSS projects, as well as getting involved and contributing back in existing ones. In the projects which Sony Mobile has released themselves, they further show that they are open for contributions by others. In the Gerrit-trigger plug-in for example, they only represent 65% of the total commits. This also gives a clear picture of the help gained by the external workforce as highlighted by OI. By opening up the Gerrit-trigger plugin and making it a part of the Jenkins community, they earn benefits such as shared feature development, maintenance and quality assurance. A reason why some of the other projects have fewer external commits (e.g., PyGerrit,

Build-failure-analyzer and Team-views) may be that they are not as established and attractive for others outside Sony Mobile. A further explanation could be that Sony Mobile has not invested the time and attention needed in order to build successful communities around these projects.

6.2 Opening Up

In relation to **RQ2**, the move to Android took Sony Mobile from a closed context to an external arena for OI, recalls the description provided by Grotnes [80]. 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 [222] can be further found on the operational level of the Tools department, which freely cooperates with both known and unknown partners in the Jenkins and Gerrit communities. From the OI perspective, these activities can be seen as a number of outside-in and inside-out transactions.

The Tools department's involvement in Jenkins and Gerrit and the associated contribution process are repetitive and bidirectional. Thus, this interaction can be classified as a coupled innovation process [70]. This also complies with Grotnes' description of how an open membership renders in a coupled process, as Jenkins and Gerrit communities both are free for anyone to join, in contrast to the Android platform and its Open Handset Alliance, which is invite-only [80].

The quantitative results provide further support for the hypothesis that both established, larger corporations and small scale software organizations are involved in the development of Jenkins and Gerrit (see Table 6). Some of the small organizations are Garmin, Ostrovsky, Luksza, Codeaurora, Quelltextlich etc. This confirms findings from the existing OI literature, e.g. [87, 202] that other community players also can use these communities as external R&D resources and complimentary assets to internal R&D processes. One possible motivation for start-ups or small scale organizations to utilize external R&D is their lack of in-house R&D capabilities. Large scale software organizations exploit communities to influence not only the development direction, but also to gain a good reputation in the community as underlined by prior studies [40, 87].

Gaining a good reputation requires more than just being an active committer. Stam [202] separates between technical (e.g. commits) and social activities (e.g. organizing conferences and actively promoting the community), where the latter is needed as complementary in order to maximize the benefits gained from the former. Sony Mobile and the Tools department have evolved in this vein as they are continuously present at conferences, hackathons and in online discussions. Focused on technical activities, the Tools department have progressively moved from making small to more substantial commits. Along with the growth of commits, they have also matured in their commit strategy. As described in Section 5.2, the intent was originally to keep the Gerrit-trigger plug-in enclosed.

This form of selective revealing [86] 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. [231]. 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 [95]. 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 [87], and evangelism of open source was a great success factor, also identified in the inner source literature [135].

6.3 Determinants of openness

When discussing if something should be made open or closed (**RQ3**) in Table 1, an initial distinction within the Tools department regarding the possible four cases is made:

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

The first two may be seen as an inside-out transaction, whilst the two latter are of an outside-in character. All have their distinct considerations, but one they have in common, as described in Section 5.2, is whether Sony Mobile will benefit from it or not. Even though the transaction cost is relative low, it still needs to be prioritized against the current needs. In the case of the two former, if a feature is too specific for Sony Mobile's case it will not gain any traction, and it will be a lost opportunity cost [124].

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 [213]. Wnuk et

al. [231] reason in a similar manner in their study where they differentiate between contributing early or late to the community in regards to specific features. By going with the former strategy, one may risk losing the competitive edge, however the latter creates potentially high maintenance costs.

Sony Mobile is aware that increased mobility [33] 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 [33]. 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 [125] Sony Mobile can use their contributions to steer and influence the direction of the community.

6.4 Requirements engineering

Tracing back to **RQ5** in Table 1, the Tools department may be viewed as both a developer and an end-user, making up a source of requirements as can often be seen in Open Source Software Development (OSSD) [192]. This applies both internally (as a supplier and an administrator of the tools), and externally (as a member of the communities). From an RE perspective, they are their own stakeholder, competing with other stakeholders (members) in the Jenkins and Gerrit communities. These are important characteristics as stakeholders who are not developers are often neither identified nor considered [4]. A consequence otherwise could be that certain areas are forgotten or neglected which stands in contrast to Wnuk et al. [231] who state that adoption of OI makes identifying stakeholders' needs more manageable. Further, this brings an interesting contrast to traditional RE where non-technical stakeholders often need considerable help in expressing themselves. The RE in OI applied through OSS can be seen as quicker, light-weight and more technically oriented than traditional RE [192].

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 [40]. Due to their high commitment and good track record, Sony Mobile employees have reached a high level in the governance organization. The Tools department combines these positions in the communities together with openness in terms of helping competitors and interacting in social activities [202] (e.g. developer conferences [112]). One reason for this is to attract quiet stakeholders, both in terms of influencing the community [39], 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 [63]

between the different stakeholders, as well as with grouping can be deemed very ad-hoc, similar to OSS RE in general [192]. This correlates to the power structure and how influence may move between different stakeholders.

Social interaction between the stakeholders is stressed by Panjer et al. [178] as an important aspect to resolve conflicts and to coordinate dependencies in distributed software development projects. The Tools department's preference for live meetings over the otherwise available electronic options such as mailing lists, issue trackers and discussion boards, is due to time differences and lag in discussions that complicate implementation of larger features. Open source hackathons [193] is the preferable choice as it brings the core stakeholders together which allows for informal negotiations [63] and a live just-in-time requirements process [54], meaning that requirements are captured in a less formal matter and first fully elaborated during implementation. As highlighted in Section 5.3, feature-by-feature collaborations is also a common practice. This is also due to the ease of communication as it may be performed between two single parties. Hence, it may be concluded that communication in this type of distributed development is a critical challenge, and in this case overcome by live meetings and keeping the number of collaborators per feature low.

This use of live-meetings and social events for requirements communication and discussion, highlights the importance of being socially present in a community other than just online if a stakeholder wants to stay aware of important decisions and implementations. Another reason for the individual stakeholder is to maintain or grow its influence and position in the governance ladder. Hence, organizations might need to revise their community involvement strategy and value what their intents are in contrast to if an online presence is enough.

Another interesting reflection on the feature-by-feature collaborations is that these may be performed with different stakeholders, i.e. relations between stakeholders fluctuate depending on their respective interests. This objective and short-term way of looking at collaborations imply a need of standardized practices in a community for it to be effective.

6.5 Testing

Addressing the **RQ5** in Table 1, we noticed during interviews that both Jenkins and Gerrit focus on manual test cases. At the same time, the communities started the transformation journey towards automated testing, with the Jenkins community leading. The openness of the Tools department led them to participate in the testing part of Jenkins community and to use its influence to rally the traction towards it amongst the other stakeholders in the community. This is especially important for the Jenkins community due to the rich number of settings offered by the plug-ins.

The Gerrit community is currently following the Jenkins' community patch, as stressed by interviewee I2. With this move towards automated testing, quality assurance will hopefully become better and enable more stable releases. These

are important aspects and business drivers for the Tools department as Jenkins and Gerrit constitute the critical parts in Sony Mobile's continuous integration tool chain. From this perspective, a trend may be seen in how the different communities are becoming more professionalized in the sense that the tools make up business critical assets for many of the stakeholders in the communities, which motivates a continuous effort in risk-reduction [86, 156].

The move towards automated testing also allowed for the Tools department to contribute their internal test cases. This may be viewed as profitable from two angles. First, it reduces the internal workload and second, it secures that settings and cases specific for Sony Mobile are addressed and cared for. The test cases may to some extent be viewed as a set of informal requirements, which secure quality aspects in regards to scalability for example which is important for Sony Mobile [20]. Similar practices, but much more formal, are commonly used in more traditional (closed) software development environments. From a community perspective, other stakeholders benefit from this as they get the view and settings from a large environment which enable them to grow as well.

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

6.6 Innovation outcomes

In relation to **RQ4** in Table 1, the focal point of the OI theory is value creation and capture [30]. 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. [52] confirmed a limited number of measurement models, and that the existing ones neither model all innovation aspects, nor say what metric can be used to measure a certain aspect. Furthermore, existing literature is scarce in regards to how data should be gathered and used for the metrics proposed in the literature. As expected, we found that Sony Mobile does not have established mechanisms in place to measure their performance before and after the Jenkins and Gerrit introduction. However, from the qualitative data collected from the interviews we specifically looked for two types of innovations: product innovations in the tools Jenkins and Gerrit, and process innovation in Sony Mobile's product development. Other types, specifically market and organizational innovation were considered but not identified.

By taking an active part in the knowledge sharing and exchange process with communities [39, 205], 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 [52]. 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 [39]. 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 [118] *"No matter who you are, not all smart people work for you"*. Hence, it is vital to reach work force beyond organizational boundaries when innovating [30], and knowledge is still retained even if people move around inside the community.

Furthermore, these software improvements and product innovations affect the performance and quality of the continuous integration process used by Sony Mobile's product development. Continuous integration as an agile practice [14] enables early identification of integration issues as well as increases the developers' productivity and release frequency [201]. With this reasoning, as reported elsewhere [127], we deem that the product innovations captured in Jenkins and Gerrit transfer on as process innovation to Sony Mobile's product development. The main reason behind this connection is the possibility to tailor and be flexible that OSS development permits. By adapting the tool chain to the specific needs of the product development the mentioned benefits (e.g. increased build quality and performance) are achieved and waste is reduced in the form of freed up hours, which product developers and testers may spend on alternative tasks, as confirmed by Moller [149]. Reduced time to market and increased quality of products are among the visible business outcomes. However, these outcomes cannot be confirmed due to a lack of objective metrics and came up as a result of interviews.

Another process innovation, which could also be classified as an organizational innovation outcome [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 [126]) 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. [152]. 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 [152]. Since Sony Mobile is a multinational corporation with a wide spread of internal culture, organizational changes are context and challenging.

6.7 Openness of tools software vs. Proprietary software

A specific aspect of **RQ2** in Table 1 is that Sony Mobile only opens up its non-competitive tools that are not the part of the revenue stream. I3 stated that *"... Sony Mobile has learnt that even collaborating with its worst competitors does not take away their competitive advantage, rather they bring help for Sony Mobile and becomes better and better"*. This raises a discussion point of why Sony Mobile limits its openness to noncompetitive tools, despite knowing that opening up creates a win-win situation for all stakeholders involved. Furthermore, it remains an open question why the research activity related to OI in SE is low, as confirmed by the results of a mapping study performed on the area [159].

In the light of the mapping study, it would be fair to state that the SE literature lacks studies on OI [159]. 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 [212]. 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 [155]. It is also important to note that this study focuses on OI via OSS participation, which is significantly different from the situation where OI is based on open source code for the product itself (like Android or Linux). In future work we plan to explore that situation to see if there are other patterns in these OI processes.

7 Conclusions

This study focuses on OI in SE at two levels: 1) innovation incorporated into Jenkins and Gerrit as software products, and 2) how these software improvements affect process and product innovation of Sony Mobile. By keeping the development of the tools open, the in- and out-flows of knowledge between the Tools department and the OSS communities bring improvement to Sony Mobile and innovate the way how products are developed. This type of openness should be separated from the cases where OSS is used as a basis for the organization's product or service offering, e.g. as a platform, component or full product [213]. To the best of our knowledge, no study has yet focused on the former version, which highlights the contribution of this study and the need for future research of the area.

Our findings suggest that both incumbents and many small scale organizations are involved in the development of Jenkins and Gerrit (**RQ1**). Sony Mobile may be considered as one of the top committers in the development of the two tools. The main trigger behind adopting OI turned out to be a paradigm shift, moving to an open source product platform **(RQ2)**. Sony Mobile's opening up process is limited to the tools that are non-competitive and non-pecuniary. Furthermore,

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

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

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

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

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

APPENDIX 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, others...)

- How do you manage and implement the requirements/features?
- How are the requirements approved and prioritized? (By developers alone, pm's, community...)
- 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, management...)
- 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 - Firm level, individual level)

Ending remarks

CHAPTER II

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

Johan Linåker, Hussan Munir, Krzysztof Wnuk and Carl Eric Mols.

Abstract

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

1 Introduction

Open Innovation (OI) has attracted scholarly interest from a wide range of disciplines since its introduction [221], but remains generally unexplored in software engineering [159]. A notable exception is that of Open Source Software (OSS) ecosystems [97, 219, 222]. Directly or indirectly adopting OSS as part of a firm's business model [32] may help the firm to accelerate its internal innovation process [30]. 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 [205, 213]. A further potential benefit is the inflow of features from the OSS ecosystem. This phenomenon is explained by Joy's law as *"no matter who you are, not all smart people work for you"*.

From an industry perspective, these benefits are highlighted in a recent study of 489 projects from European organizations that showed projects of organizations involving OI achieved a better financial return on investment compared to organizations that did not involve OI [49]. Further, two other studies [121, 157] have shown that organizations with more sources of external knowledge achieved better product and process innovation for organization's proprietary products. Moreover, a recent survey study [34] in 125 large firms of EU and US showed that 78% of organizations in the survey are practicing OI and neither of them has abandoned it since the introduction of OI in the organization. This intense practicing of OI also leads 82% of the organizations to increase management support for it and 53% of the organizations to designate more than 5 employees working full-time with OI. Moreover, the evidence suggests that 61% of the organizations have increased the financial investment and 22% have increased the financial investment by 50% in OI.

To better realize the potential benefits of OI resulting from participation in OSS ecosystems, firms need to establish synchronization mechanisms between their product strategy and product planning [64], and how they participate in the ecosystems and position themselves in the ecosystem governance structures [10, 159, 202, 231]. This primarily concerns firms that either base their products on OSS or employ OSS as part of their sourcing strategy. To achieve this synchronization, these firms need to enrich their product planning and definition activities with a strategic perspective that involves what to keep closed and what to contribute as OSS. We label this type of synchronization as *strategic product planning* in OI. *Contribution strategies* [231], i.e., guidelines that explain *what* should be contributed, and *when* play a vital role here. A common strategy is to contribute parts considered as a commodity while keeping differentiating parts closed [86, 219]. The timing aspect is critical as functionality sooner or later will pass over from being differentiating to commodity due to a constantly progressing technology life-cycle [212]. This strategy is further emphasized by existing commoditization models [22, 212]. However, these models are not designed with active OSS ecosystem participation

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

In this paper, we occupy this research gap by presenting a Contribution Acceptance Process (CAP) model. The model was developed in close collaboration with Sony Mobile. Sony Mobile is actively involved in a number of OSS ecosystem, both in regard to their products features and their internal development infrastructure[1]. With the consideration of OSS as an external asset, the CAP model is based on the Kraljic's portfolio purchasing model which helps firms analyze risk and maximize profit when sourcing material for their product manufacturing [116]. The original model is adapted through an extensive investigation of Sony Mobile's contribution processes and policies, and designed to support firms' strategic product planning. More specifically, the model helps firms to create contribution strategies for their products and software artifacts such as features and components. Hence, the CAP model is an important step for firms that use OSS ecosystems in their product development and want to gain or increase the OI benefits, such as increased innovation and reduced time-to-market. Moreover, we help firms to operationalize the CAP model by proposing an information meta-model. The meta-model is an information support that should be integrated into the requirements management infrastructure and enables contribution strategies to be communicated and followed up on a software artifact-level throughout a firm's development organization. As a first validation outside of Sony Mobile, the CAP model was presented to and applied in three case firms. This provided understanding of the model's generalizability, and also input to future design cycles.

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

2 Related work

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

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

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

2.1 Open Innovation in Software Engineering

OI is commonly explained by a funnel model [30] representing a firm's R&D process, see Fig. 1. The funnel (1) is permeable, meaning that the firm can interact with the open environment surrounding it. This conceptualization fits onto many contexts, e.g., a firm that takes part in a joint-venture or start-up acquisition. In our case, we focus on ecosystems (2) and specifically those based on OSS [68, 97]. An OSS ecosystem consists of the focal firm along with other actors who jointly see to the development and maintenance of an OSS project, which may be seen as the technological platform underpinning the relationships between the actors [100, 141]. In the context of this study, the focal firm represented by the OI funnel is Sony Mobile and their internal software development process. The OSS ecosystem could, for example, be represented by that surrounding the Android Open Source Project[2] (AOSP). The interactions between the focal firm and the ecosystem (see Fig. 1) are represented by the arrows going in and out and can be further characterized as knowledge exchange between the firm and the OSS ecosystem (e.g., Sony Mobile and AOSP). Examples of transactions can include software artifacts (e.g., bug fixes, features, plug-ins, or complete projects), but also opinions, knowledge, and support that could affect any step of the internal or external development.

The interactions (3) may be bi-directional in the sense that they can go into the development process from the open environment (*outside-in*), or from the development process out to the open environment (*inside-out*). *Coupled innovation* [53] happens when outside-in and inside-out transactions occurs together (i.e., consumption of and contribution to OSS). This may be expected in co-development between a firm and other ecosystem participants in regard to specific functionality (e.g., Sony Mobile's developer toolkits[3]).

How firms choose to work with and leverage these interactions with OSS ecosystems impact how they will realize the potential benefits of OI, such as increased innovation, shorter time-to-market, and better resource allocation [205, 213]. The CAP model presented in this paper provides operational and decision-making guidelines for these firms in terms what they should contribute to and source of from the OSS ecosystems. I.e., how they should interact with the open environment in an inside-out, outside-in, or coupled direction. Hence, what the CAP model brings in terms of novelty is an operational OI perspective on what firms involved in OSS ecosystems should share, by helping firms motivate the contributions through the creation of tailored contribution strategies.

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

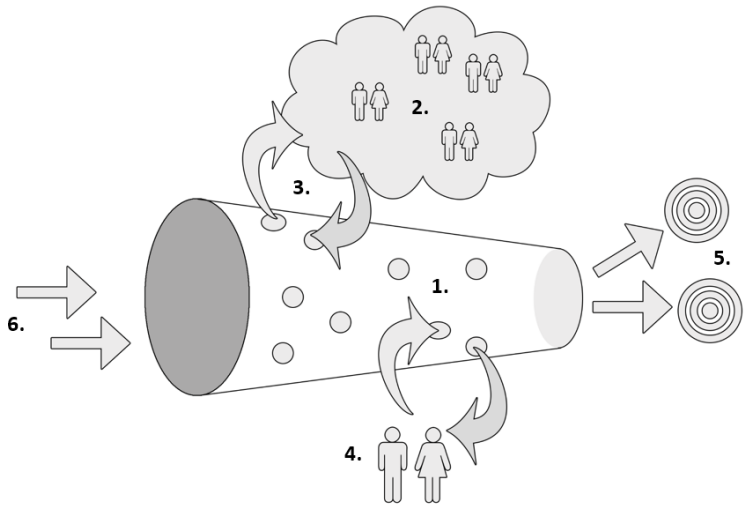


Figure 1: The OI model illustrated with interactions between the firm (1) and its external collaborations (2,4). Adopted from Chesbrough [30].

2.2 Contribution Strategies in Open Source Software Ecosystem

Wnuk et al. [231] define a contribution strategy as a managerial practice that helps to decide what to contribute as OSS, and when. To know what to contribute, it is important for firms to understand how they participate in various OSS ecosystems in regards to their business model and product strategy from an OI perspective. Dahlander & Magnusson [39] describe how a firm may access the OSS ecosystems in order to extend its resource base and align its product strategy with ecosystems' strategies. In another study, Dahlander & Magnusson [38] describe how a firm can adapt its relationships with the OSS ecosystems based on how much influence the firm needs, e.g., by openly contributing back to the OSS ecosystem, or by keeping new features internal. To build and regulate these relationships, a firm can apply different revealing strategies in this regard: differentiating parts are kept internal while commodity parts are contributed [86, 219]. Further, licenses may be used so that the technology can be disclosed under conditions where control is still maintained [219]. Depending on the revealing strategy the level of openness may vary from completely open, partly transparent conditions [32], to completely closed. As highlighted by Jansen et al. [98], the openness of a firm should be considered as a continuum rather than a binary choice.

2.3 Commoditization Models

With commoditization models, we refer to models that describe a software artifact's value depreciation [108] and how it moves between a differential to a commodity state, i.e., to what extent the artifact is considered to help distinguish the focal firm's product offering relative to its competitors. Such models can help firms better understand what they should contribute to OSS ecosystems, and when, i.e., provide a base to design contribution strategies [231]. Van der Linden et al. [212] stressed that efficient software development should focus *"... on producing only the differentiating parts"* and that *"... preferably, firms acquire the commodity software elsewhere, through a distributed development and external software such as [commercial software] or OSS"*. Firms should hence set the differentiating value of a software artifact in relation to how it should be developed, or even if it should be acquired. Commoditization is also related to the product's life-cycle and, is more often experienced towards the end of the life cycle [110].

Van der Linden et al. [212] present a commoditization model that highlights how commoditization is a continuous and inevitable process for all software artifacts. Therefore, firms should consider whether the software or technology should be developed, acquired, or kept internally, shared with other firms, or made completely open (e.g., as OSS) [11]. Ideally, differentiating software or technology should be kept internal, but as their life-cycle progresses their value depreciates and they should be made open. This is particularly relevant for software artifacts

that have an enabling role for cross-value creation, data collection or support value creation when combined with other parts of the offering, e.g., an artifact that collects and analyzes anonymous customer data that could be offered as business intelligence to customers [108]. Bosch [22] presents a similar commoditization model, which classifies the software into three layers and describes how a software's functionality moves from an early development stage as experimental and innovative, to a more mature stage where it provides special value to customers and advantage towards competition, then finally transitioning to stage where it is considered as commodity, hence it *"... no longer adds any real value"* [22].

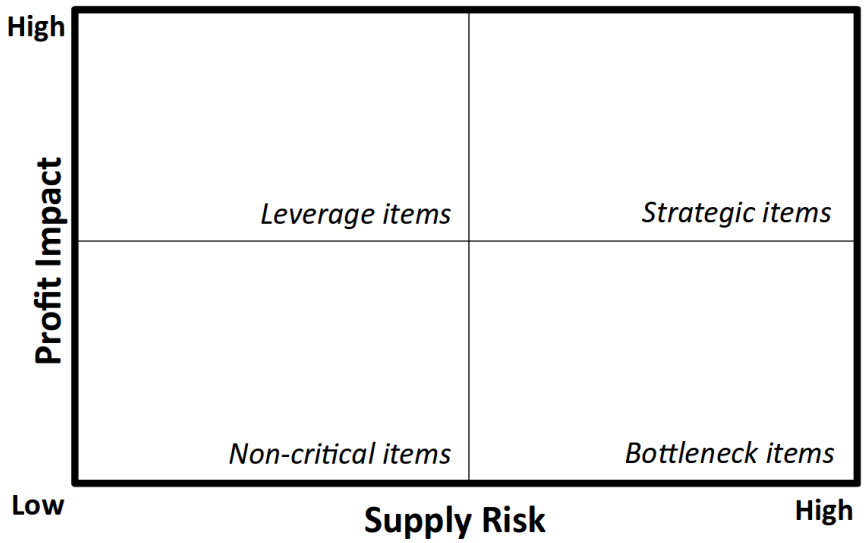
A challenge identified by both van der Linden et al. [212] and Bosch [22] is the risk of losing Intellectual property rights (IPR) to competitors, a challenge that has also been highlighted in numerous other studies [86, 87, 222, 231]. By not contributing software and technology that are considered differentiating, firms can avoid the risk of giving away its added value to competitors. However, both van der Linden et al. [212] and Bosch [22] highlight how the acquisition of the commodity functionality may help firms to reduce the development and maintenance cost, and potentially shorten time-to-market. Instead, they can shift internal focus to differential features and better-justified R&D activities [212].

2.4 The Kraljic Portfolio Purchasing Model

From the software product planning perspective, sourcing refers to decisions of what parts of the software that should be developed internally or acquired externally, from where and how [110], and is an important part of a firm's product strategy [64]. A recent literature review of software component decision-making making lists four sourcing strategies: in-house, outsourcing, COTS and OSS and brings supporting evidence that two sourcing strategies are often considered [11]. From an OSS perspective, sourcing, therefore, regards decisions on if, and what, parts of the internal software that should be based on and/or co-developed as OSS (also referred to as *Open-Sourcing* [2]). This is further highlighted in existing commoditization models (see section 2.3), which argues how commodity parts should be acquired, contributed and sourced in different ways, while internal development should be focused on differenting parts [22, 212]. With this background, we have chosen to base the CAP-model presented in this study on the portfolio purchasing model by Peter Kraljic [116].

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

Figure 2: The matrix used in Kraljic's portfolio purchasing model [116], which allows supply-items needed for a product to classified into four item categories based on the two dimensions Business impact and Supply risk.



a specific item category with its own distinctive purchasing strategy towards the suppliers [116].

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

Determining how a material or component should be classified may be done in several ways. Gelderman et al. [73] report how a consensus-seeking method is frequently used by inviting cross-functional competencies and internal stakeholders to discuss how items should be rated in regard to the two dimensions [73]. Other measurement approaches involve representing each dimension with a specific variable (e.g., supply risk as a number of available suppliers), or using a set of variable and weighting them together. After a set of items have been analyzed and put on the matrix, discussions, and reflections are performed and can potentially lead to a revision of the item categorization [73]. This discussion may concern how the firm should maintain the items' current positions or strive to move certain items between the quadrants.

The model inspired several industries and academics. Among some examples, Caniëls and Gelderman [25] studied the choice of various purchasing strategies and empirically quantified the "relative power" and "total interdependence" aspects among Dutch purchasing professionals. Ulkuniemi et al. [210] looked at purchasing as a market shaping mechanism and identified five types of market shaping actions. Shaya discussed the usage of the Kraljic's portfolio model for optimizing the process of sourcing IT and managing software licenses at Skanska ITN [199]. Gangadharan et al. proposed using Kraljic's portfolio model for mapping SaaS services and sourcing structure [67]. To the best of our knowledge, no study has suggested using Kraljic's model in the context of OSS ecosystems and creation of contribution strategies for software artifacts.

2.5 Strategic Product Planning in OI

A software product strategy defines the product and describes how it will evolve for a longer period of time [64]. It should consider aspects such as the product definition in terms of functional and quality scope, target market, delivery model, positioning and sourcing [4]. Product planning executes product strategy with the help of roadmapping, release planning, and requirements management processes [64]. Hence, decisions regarding if, and what parts of the product should be based on OSS concerns executive management and the software product management (SPM) as they usually oversee the product strategy [139], but also the development organization as they, together with SPM, oversee the product planning and development.

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

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

2.6 Artifacts in Software Engineering

The CAP model presented in this paper offers a tool for firms to decide whether or not a software artifact should be contributed to an OSS ecosystem or not. In this context, a software artifact may refer to a functionality of different abstractions, e.g., bug-fixes, requirements, features, architectural assets or components. These artifacts may be represented and linked together in software artifact repositories [57], often used for gathering, specification and communication of requirements inside a software development organization's requirements management infrastructure [15].

Artifacts may be structured and stored in different ways depending on the context and process used [57]. The resulting artifact structure (also called infrastructure) supports communication between different roles and departments inside an organization, e.g., to which product platform a certain feature belongs, what requirements a certain feature consists of, what test cases that belong to a certain requirement, which release a certain requirement should be implemented in, or what artifacts patches that represent the implementation of a certain requirement. The communication schema should be altered dependent on the firms' needs and processes [56], e.g. to follow-up what requirements are contributed. In this study, we introduce an information meta-model that proposes how a set of repositories may be set up to support the above-mentioned communication and decision-making.

Firms often store software artifacts in a central database and require certain quality criteria in terms of completeness and traceability etc [4]. In contrast, OSS ecosystems constitute an opposite extreme with their usually very informal practices [54]. Here, a requirement may be represented by several artifacts, often complementing each other to give a more complete picture, e.g., as an issue, in a mail thread, and/or as a prototype or a finished implementation. These artifacts are examples of what Scacchi refers to as informalisms [192] and are stored in decentralized repositories (such as issue trackers, mailing lists, and source code repositories respectively).

2.7 Summary

Software engineering has received limited attention in the context of OI, specifically in relation to OSS, which is widespread in practice [159]. Hence, the limited attention that contribution strategies have gotten is not surprising with some exceptions [202, 231]. There is literature explaining general incentives and strategies for how firms should act [39, 88, 222], but neither of the aforementioned or existing models [22, 212] consider aspects specific to OSS, and how firms should synchronize internal product strategy and planning with OSS ecosystem participation [159]. This study aims to address this research gap through a close academia and industry collaboration.

3 Research methodology

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

3.1 Case Firm

Sony Mobile is a multinational firm with roughly 5,000 employees, developing mobile phones and tablets. The studied branch is focused on developing Android based phones and tablets and has 1600 employees, of which 900 are directly involved in software development. Sony Mobile develops software using agile methodologies and uses software product line management with a database of more than 20,000 features suggested or implemented across all product lines [180].

As reported in earlier work [157], Sony Mobile is a mature OSS player with involvement in several OSS projects. Their existing processes for managing contribution strategies and compliance issues is centrally managed by an internal group referred to as their OSS governance board [157] (cf. OSS Working group [107]). The board has a cross-functional composition as previously suggested with engineers, business managers, and legal experts, and applies the reactive approach as described in section 4.3.

3.2 Research Questions

This study aims to support software-intensive firms involved in OSS ecosystems with integrating their internal product strategy and planning [64] with the decision-process of what software artifacts that they should contribute to the OSS ecosystems, and when, formalized as contribution strategies [231]. Strategic product planning in OI primarily concerns what parts should be revealed (contributed) in an inside-out direction [30] from the firm to the ecosystem. This contribution affects the OSS which in turn is sourced in an outside-in direction [30] from the ecosystem to the firm and is a key enabler in achieving the potential benefits of OI [159]. Earlier research in this area of OI [221], and OSS [159], is sparse and often limited to a management level (e.g., [38, 39, 86, 212]). To occupy this research gap, we aim to design a solution that supports firms in strategic product planning. We pose our first research question (**RQ1**) as:

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

Product planning is a broad practice and usually involves a cross-functional set of internal stakeholders (e.g., legal, marketing, product management, and developers) [114]. This is also the case for strategic product planning and associated

contribution strategies. For a firm with a small development organization, these internal stakeholders may be co-located and efficiently communicate and discuss decisions on a daily basis, but for larger (geographically-distributed) development organizations this may not be possible and cumbersome [41]. A contribution strategy for a certain feature needs to be communicated from the product planning team to the development teams who should implement and contribute accordingly. Conversely, product planning is responsible for monitoring the realization of the approved contribution strategies and what impact they have.

One of the main challenges for market-driven firms is to know what requirements-associated information to obtain, store, manage, and how to enable efficient communication across all stakeholders involved in the crucial decisions that lead to product success [106, 182]. Handling information overload [232] and efficiently connecting the necessary bits and pieces of information is important for strategy realization and follow up analysis. This is particularly important when introducing new concepts that require close collaboration and efficient communication between product management and product development organizations. Thus, RQ2 focuses on the information meta-model that should be integrated into the software artifact repositories used for requirements management and product planning. Our goal is to develop an information meta-model that describes how contributions to OSS ecosystems can be traced to internal product requirements and platforms, and vice versa, and allow for an organizational adoption of contribution strategies for concerned firms. This leads us to pose our second research question (**RQ2**):

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

By answering these two research questions our goal is to create a practical solution for uncovering further benefits that OI brings [159].

3.3 Research Design and Operation

This study is a design science [90] inspired case study [191]. The work was initiated by problem identification and analysis of its relevance. This was followed by an artifact design process where the artifacts (the CAP model and information meta-model) addressing the research problems (**RQ1 & RQ2**) was created. Finally, the artifacts were validated in the context of the research problem. These steps were performed in close academia-industry collaboration between the researchers and Sony Mobile. We performed data collection and analysis throughout the steps and concluded with reporting of the results (see Fig. 3).

Problem Identification

The objectives of the problem investigation phase in the design process [90] are to further understand the problem context and current practices. To gain greater understanding, we conducted informal consultations with four experts (I1-I4) at Sony Mobile who is involved in the decision-making process of OSS contributions (see Table 1). This allowed us to further refine both **RQ1** and **RQ2** and confirmed their importance and relevance for the industry. Simultaneously, internal processes and policy documentation at Sony Mobile were studied. Next, we received permission to access additional data sources and were able to investigate requirements and contribution repositories. The consultations and investigations confirmed that a suitable solution requires a combination of a technology-based artifact and an organization-based artifact (see guidelines one and two by Hevner [90]). The technology-based artifact (**RQ1**) should allow firms to create contribution strategies for software artifacts and the organizational-based artifact (**RQ2**) should support the organizational adoption and operationalization of the technology-based artifact.

Table 1: Consultation with experts

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

Artifact Design

RQ1 is addressed by designing an artifact that would allow the practitioners to decide whether a software artifact should be contributed to an OSS ecosystem or not. As this is a sourcing issue at the product strategy-level [11, 64, 110], we decided to base the artifact on Kraljic's portfolio purchasing model [116] following the advice and experience of I2 in sourcing. The model consists of a matrix that allows firms to analyze how they source and purchase material and components for their production (see section 2.4).

With this foundation, we iteratively formalized our findings from the consultations with I1-I4 and studies of internal processes and policy documentation. The results of this formalization are the CAP model and the associated meta-model of information required to instantiate the CAP model, supporting strategic product planning in OI. Each item category from the original model [116] has a corresponding type of contribution strategy [231], and instead of supply items, we refer to software artifacts, e.g., features or components. The two dimensions are re-

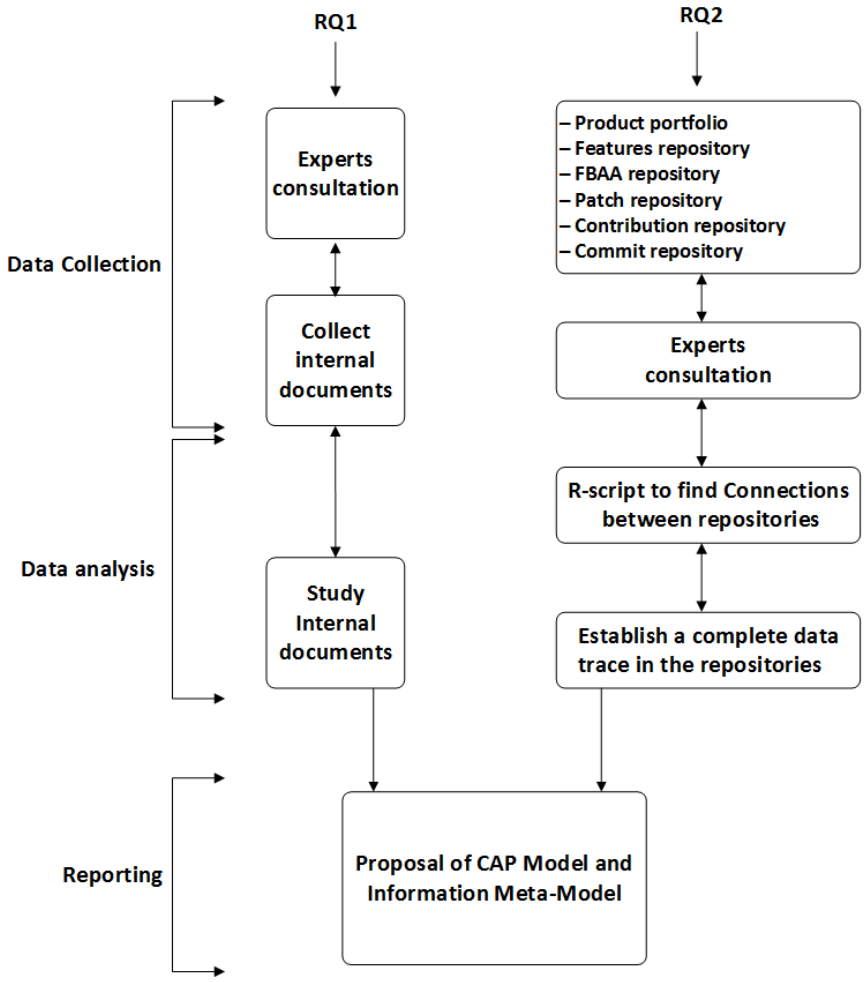


Figure 3: Overview of the research methodology used in this study. The design process was performed iteratively through the three steps involved: problem investigation, artifact design, and artifact valuation [90].

fined to represent *Business impact* and *Control complexity*, inspired by existing commoditization models [22, 212] and literature on OSS ecosystem governance (e.g., [10, 38, 161]). The measurement process is proposed to employ a consensus-seeking approach [73] with the involvement of cross-functional competencies and internal stakeholders [114]. To help frame the measurement discussion process, questions are defined inspired by literature related to the Kraljic portfolio purchasing model (e.g., [25, 73]), commoditization models [22, 212], software value map (e.g., [9, 108], and OSS ecosystem governance (e.g., [10, 38, 161]). An overlay is created on top of the CAP model to highlight which contribution objective should be the primary driver for the chosen contribution strategy. The objectives represent important value incentives inspired by OI literature [30, 205, 213, 219]. The intention is to help users of the model to fine-tune the contribution strategy for the classified artifact. The CAP model is presented in more detail in section 4.

To address **RQ2** and enable an organizational adoption and operationalization of the CAP-model, we created an information meta-model that facilitates communication and follow-up on software artifacts and their contribution strategies. In the problem investigation phase, it became apparent that the information support should be integrated into the software artifact repositories used for requirements management. The information support would then be able to reach everyone who is involved in the product planning and development. This required us to expand our investigation of Sony Mobile's requirements and contribution repositories, which included a broad set of software artifact repositories that are used in the product planning of mobile phones. We focused the repository investigation on understanding how contributions could be traced to product requirements and platforms, and vice versa. Through consultation with I1-I4, we selected six relevant repositories: the internal product portfolio, feature repository, feature-based architectural asset repository, patch repository, contribution repository and commit repository (see section 5).

These repositories and their unique artifact IDs (e.g., requirement id, patch id, and contribution id) allowed us to trace the contributions and commits to the architectural assets, product requirements and platforms, via the patches that developers create and commit to internal source code branches. This analysis resulted in the information meta-model presented in Fig. 5. The meta-model creation process was driven by the principles of finding a balance between research rigor and relevance, moving away from extensive mathematical formalizations of the CAP model and focusing on the applicability and generalizability of the model, see guideline five by Hevner [90].

Artifacts Validation

Validation helps confirm that candidate solutions actually address the identified research problems. As we are in an early stage of the research and design process, this study uses static validation [79]. This type of validation uses presentation of

candidate solutions to industry practitioners and gathering of feedback that can help to further understand the problem context and refine candidate solutions, in line with the design science process [90]. Dynamic validation [79], which concerns piloting of the candidate solutions in a real-work setting, is a later step in the technology transfer process and is currently under planning at the case firm and is left for future work.

Both the CAP-model and its related information meta-model were validated statically through continuous consultations with experts at Sony Mobile (I1-I4). In these consultations, the models were explained and discussed. Feedback and improvement ideas were collected and used for iterative refinement and improvement. Experts were asked to run the CAP model against examples of features in relation to the four software artifact categories and related contribution strategies that CAP model describes. The examples are presented together with the CAP model and provide further detail and validation of its potential use, see section 4.4. A complete example of how the CAP model and meta-model are used is further presented in section 6. These examples help to evaluate functionality, completeness, and consistency of the CAP model and associated information meta-model. The usability of the information meta-model was further validated by performing traces between the different types of artifacts and their repositories. These traces were presented and used in the static validation of the meta-model. From a design science perspective [90], we employed observational validation through a case study at Sony Mobile where we studied the artifacts (models) in a business environment. We also employed descriptive evaluation where we obtained detailed scenarios to demonstrate the utility of the CAP model, see guideline three by Hevner [90].

To improve the external validity of the CAP model, we conducted exploratory case studies at three different case firms (see Section 7). In these case studies, we used static validation [79] where we presented the CAP model to participants from the respective firms and applied it in a simulated setting as part of the interviews. In two of the cases, semi-structured interviews were used with one representative from each firm. In the third case, a workshop setting was used with eight participants from the firm. When collecting feedback from the three case firms, we focused on applicability and usability of the CAP model.

3.4 Ethics and Confidentiality

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

Sony Mobile who reviewed the manuscript to ensure the validity and transparency of results for the scientific community.

3.5 Validity Threats

This section highlights the validity threats associated with the study. Four types of validity threats [191] are mentioned along with their mitigation strategies.

Internal Validity

Internal validity refers to factors affecting the outcome of the study without the knowledge of the researchers [191].

Researcher bias refers to when the researcher may risk influencing the results in a wanted direction [190]. The proposed CAP model was created with an iterative cooperation between researchers and industry practitioners. Thus, there was a risk of introducing the researcher's bias while working towards the creation of the model. In order to minimize this risk, regular meetings were arranged between researchers and industry experts to ensure the objective understanding and proposed outcomes of the study. Furthermore, researchers and industry practitioners reviewed the paper independently to avoid introducing researcher's bias.

A central part of the CAP model involves estimating the business impact and control complexity. These estimations involve several factors and can have multiple confounding factors that influence them. In this work, we assume that this threat to internal validity is taken into consideration during the estimation process and therefore is not in the direct focus of the CAP model. Moreover, the CAP model does not prevent additions of new factors that support these estimates.

Triangulation refers to the use of data from multiple sources and also ensuring observer triangulation [190]. In this study, our data analysis involved interpretation of qualitative and quantitative data obtained from Sony Mobile. We applied data triangulation by using Sony Mobile's internal artifacts repositories, documents related to contribution strategies and consultation with relevant experts before proposing the CAP model. There were risks of identifying the wrong data flows and subjective interpretation of interviews. In order to mitigate these risks, concerned multiple experts with different roles and experiences (see Table 1) were consulted at Sony Mobile. We ensured observer triangulation by involving all researchers who authored this manuscript into the data collection and analysis phases.

External Validity

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

We have focused on analytic generalization rather than statistical generalization [59] by comparing the characteristics of the case to a possible target and presenting case firm characteristics as much as confidentiality concerns allowed. The scope of this study is limited to firms realizing OI with OSS ecosystems. Sony Mobile represents an organization with a focus on software development for embedded devices. However, the practices that are reported and proposed in the study has the potential to be generalized to all firms involved in OSS ecosystems. It should be noted that the case firm can be considered a mature firm in relation to OSS usage for creating product value and realizing product strategies. Also, they recognize the need to invest resources in the ecosystems by contributing back in order to be able to influence and control in accordance with internal needs and incentives. Thus, the application of the proposed CAP model in an other context or in other firms remains part of future work.

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

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

Construct Validity

Construct validity deals with choosing the suitable measures for the concepts under study [191]. Four threats to the construct validity of the study are highlighted below.

First, there was a risk that academic researchers and industry practitioners may use different terms and have different theoretical frames of reference when addressing contribution strategies. Furthermore, the presence of researchers may

have biased the experts from Sony Mobile to give information according to researchers' expectations. The selection of a smaller number of experts from Sony Mobile might also contribute to the unbalanced view of the construct.

Second, there was a potential threat to construct validity due to the used innovation assessment criteria based on business impact and control complexity. Both dimensions can be expanded by additional questions (e.g. internal business perspective or innovation and learning perspective [108]) and the CAP model provides this flexibility. One could argue that also technical and architectural aspects should be taken into consideration here. At the same time, the static validation results at Sony Mobile confirm that these aspects have limited importance at least for the studied cases. Still, they should not be overlooked when executing the CAP model in other contexts.

Third, a common theoretical frame of reference is important to avoid misinterpretations between researchers and practitioners [190]. In this study, the Kraljic's portfolio model is used as a reference framework to the CAP model. However, the horizontal and vertical dimensions of Kraljic's portfolio model were changed to control complexity and business impact respectively. Both industry practitioners and academic researchers had a common understanding of Kraljic's portfolio model [116] before discussions in the study. Furthermore, theoretical constructs were validated by involving one of the experts in the writing process from Sony Mobile to ensure consistent understanding.

Fourth, prolonged involvement refers to a long-term relationship or involvement between the researchers and organization [190]. Since there was an involvement of confidential information in the study, it was important to have a mutual trust between academic researchers and practitioners to be able to constructively present the findings. The adequate level of trust was gained as a result of long past history of collaboration between academic researchers and experts from Sony Mobile.

Reliability

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

Member checking may involve having multiple individuals go through the data, or letting interviewees review a transcript [190]. In this study, the first two authors proposed the meta-model after independent discussions and reviewed by the third author. Furthermore, the model was validated by a team lead, software developer, and senior manager at Sony Mobile, involved in making contributions to OSS communities, were consulted to ensure the correctness of the meta-model and associated data.

Audit trail regards maintaining traceability between collected data during the study [190]. For this study, the first two researchers kept track of all the mined data from the software artifact repositories as well as the email and informal communi-

cation between researchers and Sony Mobile representative. Results were shared with Sony Mobile for any possible misinterpretation or correction of data.

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

The CAP model is an adapted version of the portfolio model introduced by Peter Kraljic [116]. Kraljic's model was originally constructed to help firms with creating purchasing strategies towards their suppliers of items needed for their product manufacturing. The CAP model is focused on software artifacts and how these should be sourced and contributed as OSS. The artifacts may be of different abstraction levels, e.g., ranging from specific requirements or issues to sets of requirements as features, frameworks, tools or higher level components.

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

4.1 Model Description

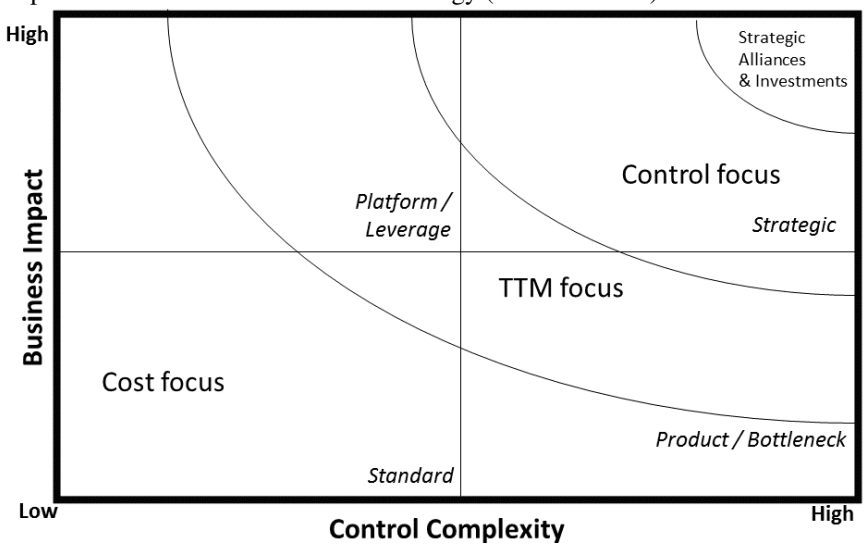
The focal point of the CAP model is the matrix presented in Fig. 4. Artifacts are mapped on to the matrix based on how they are valued in regard to the two dimensions *Business impact* and *Control complexity*, located on the vertical and horizontal axis respectively. Business impact refers to how much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

Artifact Types and Contribution Strategies

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

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

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



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

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

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

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

ply that the artifact will be problematic to contribute in a general OSS ecosystem. An option would then be to identify and target specific stakeholders of interest, i.e. of customers and their suppliers, and create a limited project and related OSS ecosystem. For examples of Products/Bottlenecks artifacts, see section 4.4.

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

Contribution Objectives

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

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

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

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

Time-To-Market (TTM) focus: Artifacts that have higher levels of competitive advantage, and/or require a higher amount of control and understanding than commodity artifacts should likely to have the general objective to be advanced to

the marketplace as soon as possible, superseding the objective of reducing maintenance costs. These artifacts may also be referred to as *qualifiers*, i.e., artifacts that are essential but still non-differential, and should be contributed as soon and often as possible in order to allow for the own solution to be established as the leading open solution. This will potentially give the advantage of control and barring competing solutions which would otherwise require additional patching or even costly redesigns to one's own product.

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

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

Adapting Contribution Strategies with Contribution Objectives

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

4.2 Proactive Approach

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

S1 Decision on scope and abstraction level.

S2 Classification and mapping artifacts to the matrix.

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

(b) Synchronize and reiterate mapping.

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

S3 Reiteration of the artifact mapping.

S4 Documentation and communication of the decisions.

S5 Monitoring and follow-up on the decisions.

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

The collected information should then be used as input to an open consensus-seeking discussion forum (**S2**), where relevant internal stakeholders help to classify the artifacts. As in the roadmapping process [114], these stakeholders should bring cross-functional perspective to the decision-making to further explain and argue based on the collected background information, e.g., representatives from marketing, product management, development, and legal.

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

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

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

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

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

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

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

4.3 Reactive Approach

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

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

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

- **Trivial contributions** are rather small changes to already existing OSS ecosystems, which enhances the non-significant code quality without adding any new functionality to the system e.g., bug fixes, re-factoring etc.
- **Medium contributions** entails both substantially changed functionality, and completely new functionality e.g., new features, architectural changes etc.
- **Major contributions** are comprised of substantial amounts of code, with significant value in regard to IPR. These contributions are a result of a significant amount of internal development efforts. At Sony Mobile, one example of such a contribution is the Jenkins-Gerrit-trigger plug-in [157].

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

4.4 Contribution Strategies with Artifact Examples

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

Strategic Artifacts:

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

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

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

Platform/Leverage Artifacts

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

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

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

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

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

Products/Bottleneck Artifacts

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

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

Standard Artifacts

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

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

[7] https://github.com/resin-io/resin-wifi-connect

[8] https://github.com/nostra13/Android-Universal-Image-Loader

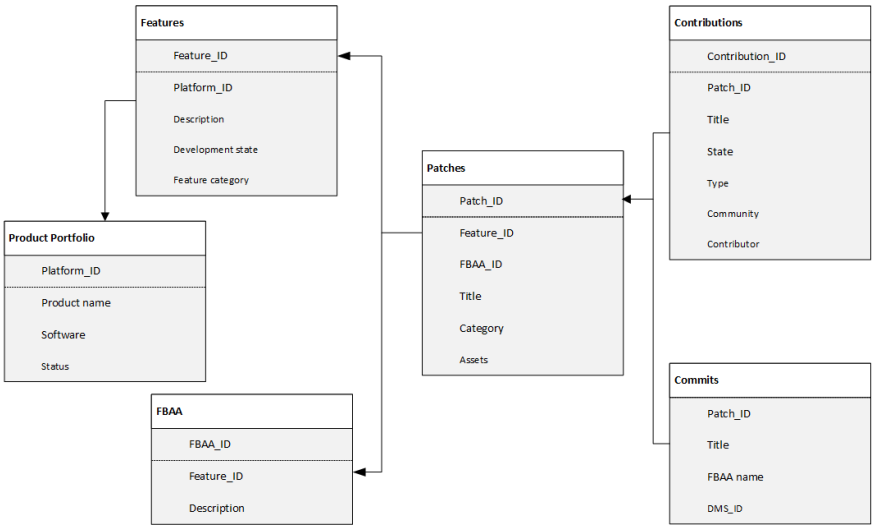


Figure 5: Software artifact repositories necessary to communicate and follow-up on contribution strategies decided with the CAP model.

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

5 Operationalization of the CAP model (RQ2)

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

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

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

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

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

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

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

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

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

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

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

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

6 Combining the CAP Model and the Information Meta-model

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

For Sony Mobile, FBAAs offer a suitable abstraction level to determine whether certain functionality (e.g., a media player or power saving functionality) can be contributed or not. If the artifact is too fine-grained it may be hard to quantify its business impact and control complexity. In these cases, features included in a certain FBAA would inherit the decision of whether it can be contributed or not. Regarding the scope, we look at FBAAs related to the telephony part of a certain platform-range. The FBAA that we classify regards the support for Voice over Long-Term Evolution (VoLTE), which is a standard for voice service in the LTE mobile radio system [181]. Note that this classification is performed when VoLTE was relatively new to the market in 2015.

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

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

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

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

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

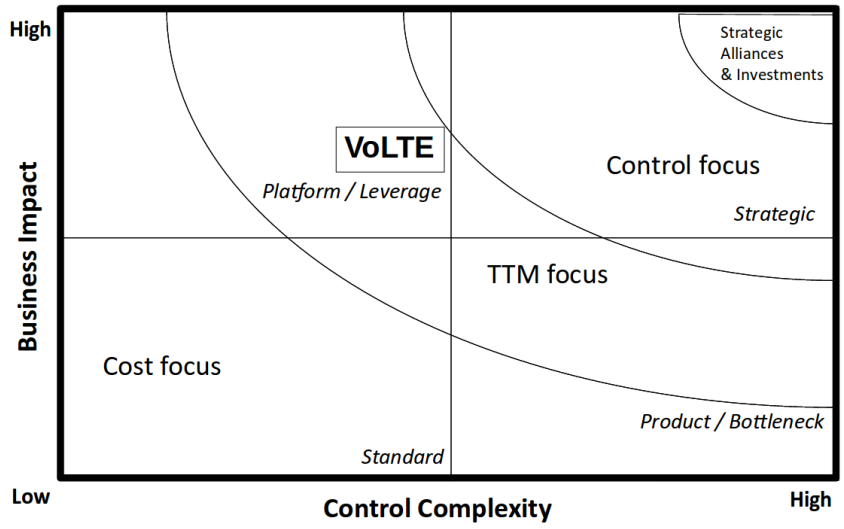


Figure 6: The CAP model and the example of VoLTE which is classified in regard to its business impact and control complexity.

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

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

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

7 Case studies

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

7.1 Case Firm A

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

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

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

Overall Contribution Strategy

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

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

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

Moreover, it may take a substantial amount of time before an intellectual property asset reveals its true value in the market place due to delays in the technology adoption curve. Therefore, cautious evaluation of the business and revenue values are necessary. If the technology adoption is slow, it is much more challenging and harder to see if and when to contribute.

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

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

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

Application of the CAP Model

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

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

- **Platform/Leverage artifacts** - Covers about 55% of all features because complexity is low and the firm has high control in case the firm becomes dominant in the market (they are currently not dominant).

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

Usability of the CAP Model

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

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

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

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

7.2 Case Firm B

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

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

Overall Contribution Strategy

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

Analyzing this market segment with the help of the CAP model brings forward how extreme the risk levels are in the mobile games business. CAP works well here as a risk assessment tool that should be applied to investments. In this market place, the quadrants of the CAP model can be merged and discussed together. The

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

main analysis should be along the Y-axis and the discussion should be profit driven since the firm does not have any control over the platform, but controls the player experience.

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

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

Application of the CAP Model

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

Usability of the CAP Model

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

The CAP model, in this case, can be used once and the clear conclusion for the firm is that it should change its market focus. The model clearly points out that if a firm is relatively new to mobile game development there is little profitability in this market unless you have 20-30 million dollars to invest in marketing and other actions to sustain long terms revenues. Our respondent believes that every new game concept can be and should be evaluated with the help of the CAP model.

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

7.3 Case Firm C

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

Overall Contribution Strategy

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

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

Application of the CAP Model

The studied project was first discussed in terms of its background and functionality in order to synchronize the knowledge level among the workshop participants. This was followed by a discussion of the project's business impact. The questions

outlined in Section 4.2 were used for framing the discussion, but instead of using the Likert scale of 1-4, the workshop participants opted for an open consensus-seeking discussion from start.

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

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

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

Usability of the CAP Model

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

The workshop-participants preferred the open consensus-seeking discussions as a mean to determine the business impact and control complexity, and based on this classify the artifact to the most relevant artifact type and contribution strategy.

The chosen strategy and aligning contribution objective could then be used to add further depth and understanding to the discussion, which helped the group to arrive at a common decision and final contribution strategy for the reviewed project.

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

8 Discussion

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

8.1 Applicability and Usability of the CAP Model

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

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

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

highlights that the decision process should not be "over-engineered". Our results suggest that complexity needs to be balanced in order to maintain usability for the practitioners while still keeping the applicability on different types of artifacts and settings. How to adapt this balancing act and tailor the CAP model to different settings is a topic for future design cycles and case evaluations.

8.2 Influence Needed to Control

The Kraljic's portfolio model was originally used to help firms to procure or source supply-items for their product manufacturing [116]. One of the model's two decision factors is *supply risk*. To secure access to critical resources, a certain level of control is needed, e.g., having an influence on the suppliers to control the quality and future development of the supply-items. For OSS ecosystems, this translates into software engineering process control, for example in terms of how requirements and features are specified, prioritized and implemented, with the goal to have them aligned with the firm's internal product strategy.

Software artifacts with a high control complexity (e.g., the media frameworks for Sony Mobile, see section 4.4) may require special ownership control and a high level of influence in the concerned OSS ecosystems may be warranted to be able to contribute them. In cases where a firm does not posses the necessary influence, nor wish to invest the contributions and increased OSS activity [38] which may be required, an alternative strategy is to share the artifact with a smaller set of actors with similar agendas, which could include direct competitors [222]. This strategy is still in-line with the meritocracy principle as it increases the potential ecosystem influence via contributions [38]. Sharing artifacts with a limited number of ecosystem actors leaves some degree of control and lowers the maintenance cost via shared ownership [205, 213]. Further, time-to-market for all actors that received the new artifacts is substantially shortened.

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

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

8.3 Direct and Indirect Use of OSS ecosystems

The second decision factor originating from the Kraljic's model [116] is the *profit impact*. Profit generally refers to the margin between what the customer is willing to pay for the final product and what the product costs to produce. For OSS ecosystems, this translates into how much *value* a firm can offer based on the OSS, e.g. services, and how much resources the firm needs to invest into integration and differentiation activities. I.e., much of the original definitions are preserved in the CAP model and the re-labeled decision factor business impact.

Artifacts with high profit, or high business impact are differential towards competitors and add significant value to the product and service offerings of the firm [212], e.g., the gaming services for Sony Mobile, see section 4.4. Analogous, artifacts with low profit are those related to commodity artifacts shared among the competitors, e.g., Linux commodity parts, see section 4.4. This reasoning works in cases where the OSS and its ecosystem is directly involved in the product or service which focal firm offers to its customers. The customers are those who decide which product to purchase, and therefore mainly contribute in the value creation process [9]. This requires good customer-understanding to judge which artifacts are the potential differentiators that will influence the purchase decision.

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

OSS ecosystems often facilitates software engineering process innovations that later spark product innovations that increase the business impact of an artifact, e.g., if an artifact makes the development or delivery of the product to a higher quality or shorter time-to-market respectively [127]. These factors cannot be judged by marketing, but rather by the developers, architects and product managers who are involved on the technical aspects of software development and delivery. In regard to the CAP model, this indirect view of business impact may be managed by having a cross-functional mix of internal stakeholders and subject-matter experts that can help to give a complete picture of an artifact's business impact.

8.4 Comparing to Other Commoditization Models

Both commoditization models suggested by van der Linden et al. [212] and Bosch [22] consider how an artifact moves from a differential to a commoditized state. This is natural as technology and functionality matures and becomes standardized among actors on the same market or within the same OSS ecosystem. The impact of whether an artifact is to be considered differential or commodity is covered by the business impact factor of the CAP model. However, how quickly an artifact moves from one state to another is not explicitly captured by the CAP model. This

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

Relative to the level of commoditization of an artifact, the two previous models consider how the artifact should be developed and shared. Van der Linden et al. [212] suggested to internally keep the differential artifacts and gradually share them as they become commoditized through intra-organizational collaborations and finally as OSS. In the CAP model, this aligns with the control complexity factor, i.e., how much control and influence is needed in regard to the artifact.

The main novelty of the CAP model in relation to the other commoditization models [22, 212] considers OSS ecosystem participation and enables improved synchronization towards firms' product strategy and product planning, via feature selection, prioritization and finally release planning [110]. The strategic aspect covered by the CAP model uses the commoditization principle together with business impact estimates and control complexity help may firms to better benefit from potential OI benefits. Assuming the commoditization is inevitable, the CAP model helps firms to fully benefit the business potential of differential features and timely share them with OSS ecosystems for achieving lower maintenance costs. Moreover, the CAP model helps to visualize the long term consequences of keeping or contributing an internally developed software artifact (more patches and longer time-to-market as consequence). Finally, the CAP model provides guidelines for how to position in an OSS ecosystem's governance structure [10] and how to influence it [38].

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

9 Conclusion

The recent changes in software business have forced software-intensive firms to rethink and re-plan the ways of creating and sustaining competitive advantage. The advent of OSS ecosystems has accelerated value creation, shortened time-to-market and reshaped commoditization processes. Harvesting these potential benefits requires improved support for strategic product planning in terms of clear guidelines of *what to develop internally* and *what to open up*. Currently available commoditization models [22,212] accurately capture the inevitability of com-

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

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

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

CHAPTER III

A COMMUNITY STRATEGY FRAMEWORK – HOW TO OBTAIN INFLUENCE ON REQUIREMENTS IN MERITOCRATIC OPEN SOURCE SOFTWARE COMMUNITIES?

Johan Linåker, Björn Regnell and Daniela Damian.

Abstract

Context: In the Requirements Engineering (RE) process of an Open Source Software (OSS) community, an involved firm is a stakeholder among many. Conflicting agendas may create miss-alignment with the firm's internal requirements strategy. In communities with meritocratic governance or with aspects thereof, a firm has the opportunity to affect the RE process in line with their own agenda by gaining influence through active and symbiotic engagements. **Objective:** The focus of this study has been to identify what aspects that firms should consider when they assess their need of influencing the RE process in an OSS community, as well as what engagement practices that should be considered in order to gain this influence. **Method:** Using a design science approach, 21 interviews with 18 industry professionals from 12 different software-intensive firms were conducted to explore, design and validate an artifact for the problem context. **Results:** A

Community Strategy Framework (CSF) is presented to help firms create community strategies that describe if and why they need influence on the RE process in a specific (meritocratic) OSS community, and how the firm could gain it. The framework consists of aspects and engagement practices. The aspects help determine how important an OSS project and its community is from business and technical perspectives. A community perspective is used when considering the feasibility and potential in gaining influence. The engagement practices are intended as a tool-box for how a firm can engage with a community in order to build influence needed. **Conclusion:** It is concluded from interview-based validation that the proposed CSF may provide support for firms in creating and tailoring community strategies and help them to focus resources on communities that matter and gain the influence needed on their respective RE processes.

1 Introduction

Open Source Software (OSS) is for many firms today a fundamental building block for creating, delivering and supporting their product and service offerings, or internal operations [24, 39]. The development and maintenance of an OSS project are performed within a software ecosystem [99], often referred to as a community. The members of a community consist of stakeholders of the OSS project, i.e., *"...person[s] or organization[s] who influences a system's requirements or who [are] impacted by that system"* [77]. In this case, "a system" refers to the OSS project. To a firm involved in an OSS community, the Requirements Engineering (RE) process in the community is an external process where the firm is no longer the central authority, in contrast to traditional market-driven RE [182]. Instead, the firm is a stakeholder among many which may introduce conflicting agendas from other stakeholders [138, 159, 194], and a new type of power and politics than the firm might be used to [146]. Consequences may include a lack of control over what requirements that are implemented, and miss-alignment with the firm's internal RE process [39, 231]. A firm who wish to affect the RE process according to their agenda may, therefore, have to build up an influence within the community [194].

With influence, we refer to the Merriam-Webster dictionary [1] which defines it as *"the power to change or affect someone or something"*. In our context, this relates to the power of a firm to change or affect a requirement of interest in an OSS community, for example, how a requirement is specified, prioritized, and realized, both short-term in release-planning, and long-term on the road-map [74, 120, 167]. In OSS communities with a meritocratic governance structure [46, 143], either in part or in full [198], influence is gained by proving merit and earning trust and status within the community [58]. What merit constitutes depends on the context [51, 174], but is generally gained by building an active

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

and symbiotic relationship with the community where a firm dedicates resources, contributes internal requirements and actively participates in the development of the OSS [24, 38, 40, 166, 194, 206]. A meritocratic OSS community, therefore, offers an opportunity for the focal firm to influence the community's RE process according to the firm's own agenda while competing and collaborating with the other stakeholders in the community [166].

For a firm engaged in many communities, such investments may be costly if it is distributed over all communities. It may be that only a few communities are of such strategic importance to the firm, and are in a state where the firm needs to have an influence on their RE processes [39]. For a strategic community that is healthy, predictable and aligned with a firm's internal agenda, it may be that a high level of influence is not motivated [24]. Therefore, to optimize its resource utilization and investments where best needed, firms may have to assess how they could benefit from a specific OSS project and its community, and then if and how much influence that is required to reap these benefits [24]. To the best of our knowledge, there is no systematic approach to perform this kind of assessment, why we pose our first research question as:

RQ1 What aspects should a firm consider when assessing its need to influence the RE process in a meritocratic OSS community?

If a firm assesses that they need influence on the RE process in a meritocratic OSS community, the follow-up question is: what should their community engagement look like and how should they invest their resources to gain the influence needed? To the best of our knowledge, an overview on a software engineering level of what engagement practices that may be used to build influence in meritocratic OSS communities is absent (e.g., [5,60,131,159]). This gap leads us to pose our second research question:

RQ2 What practices should a firm consider to gain influence on the RE process in a meritocratic OSS community?

To address these two research questions, this paper presents a Community Strategy Framework (CSF). A community strategy should describe if and why a firm needs influence on the RE process in a specific OSS community, and how the firm could gain it. Thus, the objective of CSF is to help firms create and tailor community strategies that enable them to focus resources on communities that matter and gain the influence needed on their respective RE processes.

Using a design science approach [90, 228], we leverage a series of ten semi-structured interviews with industry professionals to explore the problem context. Interview transcripts were then inductively coded [191] which resulted in a first design of the CSF. To validate and refine the design, seven interviews were conducted where the interviewees were presented with the CSF and asked questions regarding its completeness and correctness. To evaluate the applicability and utility of CSF [90], in one of these interviews, the framework was also applied to a

fictitious example based on an earlier reported case study [157]. As the last step, a case validation was conducted by interviewing four industry professionals from a software-intensive firm engaged in multiple OSS communities. Questions focused on the validity of CSF in the context of the firm's community engagements. In total, we conducted 21 interviews with 18 industry professionals from 12 different software-intensive firms.

The rest of the paper is structured as follows: in Section 2, we present related work, which this study builds upon. In Section 3, we present the research design of this study and how it was executed. In Section 4, we present the CSF, and in Section 5 the framework is applied to a fictitious example. In Section 6 we discuss our findings, followed by a discussion on threats to validity in Section 7. In Section 8, we conclude the paper.

2 Related Work

In this section, we present the related work that provides a theoretical underpinning for the design of the artifact called the Community Strategy Framework (CSF). This theoretical basis is also used in the discussions on the validity of the proposed framework (see Section 6).

2.1 Requirements Engineering in OSS communities

Compared to classic RE [4], OSS RE can be described as a collaborative, transparent and open process involving the stakeholders (both developers and users) in the community with interest in specific requirements [2, 4]. Formal methods and processes, as well as documents or central repositories, are often absent [27, 117]. Instead, a requirement may often be represented by multiple artifacts which are stored and managed in a series of interconnected and overlapping repositories, e.g., as an issue in an issue tracker and mail threads in a mailing list [193]. These repositories also function as communication channels for the stakeholders where the requirements are asserted (i.e., elicited from the OSS community perspective), analyzed, and specified informally, and often realized simultaneously [16,27,54,117]. This is an iterative process characterized as just-in-time RE [16, 54] and where the social interactions between the stakeholders are often decentralized and dynamic [17]. However, these can on occasion also occur centralized in "off-line" events such as conferences, meet-ups, and hackathons [38, 157, 202].

Prioritization and selection of requirements are commonly performed by individuals in leadership positions of the OSS community, however, with consideration taken to expressed wishes of the community [74, 120, 167]. This hierarchy between the roles in OSS communities is often depicted with the help of an onion model [161]. In its multi-layered construction, central and leadership roles can be found among the core layers, while the passive users can be found in the outer

ones (cf. Core-Periphery Model [103]). The structure implies that the further out a community member is, the less direct influence and knowledge the person has over the project's state and direction [101]. Furthermore, what roles that exist in a community, specifically regarding leadership, may differ between communities. Some may, for example, have a project lead as with Linus Torvalds in the Linux kernel community, while some may have a core team of entrusted members as in the PostgreSQL community [161].

Migration between layers can be fluid and agile depending on the project, e.g., community members can move between multiple layers, or be recruited into one, bypassing outer ones [101]. This migration further depends on the type of governance in the community.

2.2 Governance in OSS communities

de Laat [46] describes OSS governance as different configurations, primarily based on the authority structure, i.e., the way that authority is established, distributed, and exercised, either through autocratic or democratic principles. In the former, leadership is centralized and top-down, while in the latter it is decentralized and bottom-up. Building on this distinction, De Noni et al. [47] refines the two configurations further as presented in Fig 1. Concerning communities with autocratic tendencies, they differentiate between sponsor-based and tolerant dictator-based communities. In the former, leadership is centered around the sponsoring firm(s), while in the latter it is centered around a single project leader (tolerant dictator). In regards to communities with democratic tendencies, De Noni et al. [47] separates open-source-based and collective communities. In open-source based communities leadership is characterized as institutionalized, democratic, and distributed, often inside the walls of a foundation. In collective communities, leadership is seen as collective, meritocratic, and distributed.

<table>
<tr><th colspan="5">Governance and Authority Structure Concepts and Relations</th><th>Reference</th></tr>
<tr><td colspan="3">Autocratic communities</td><td colspan="2">Democratic communities</td><td>de Laat (2007)</td></tr>
<tr><td colspan="2">Sponsor-based communities</td><td>Tolerant dictator-based communities</td><td>Open-source-based communities</td><td>Collective communities</td><td>De Noni et al. (2013)</td></tr>
<tr><td>Commercial OSS projects</td><td colspan="4">Community OSS projects</td><td>Capra and Wasserman (2008)</td></tr>
<tr><td>Single-vendor OSS projects</td><td colspan="4">Multivendor OSS projects</td><td>Schaarschmidt et al. (2015)</td></tr>
</table>

Figure 1: Overview of governance and authority structure concepts in OSS projects and their relations as presented in Section 2.

Capra and Wasserman [26] makes a distinction between commercial and community OSS. In the former, the OSS project is owned and managed by a single

firm [186], i.e., a special case of sponsor-based communities [47]. In the latter, the community is owned and managed by the community, which may include one or more firms, also aligning with the community-managed governance model as described by O'Mahony [172]. Schaarschmidt et al. [194] further label these types of projects as single-vendor projects and multivendor projects respectively.

Even with the categorizations of OSS governance models and their authority structures shown in Fig 1, other research shows that the picture can be more blurry. According to the literature review by Shaikh and Henfridsson [198], research has been consistent in describing how communities can only have one authority structure (with one notable exception [75]). Even though a community can evolve its authority structure in hybrid forms with time, a single authority structure will result in the end [174]. However, based on their view of a duality between governance and coordination, Shaikh and Henfridsson [198] move to suggest that multiple forms of authority structures can co-exist in parallel, each embedded in and operationalized by a coordination process. These coordination processes can integrate, and evolve together within a community, of which some may pass out with time and be replaced by others. In their longitudinal analysis of the Linux kernel community, they identified a varying mix of autocratic and oligarchic structures, but also semi-autonomous governing in terms of the different sub-modules. Meritocracy was continuously present through the analysis. I.e., even tolerant dictator-based communities can show traits of a community-managed [172] and meritocratic [46] governance model.

Although literature lists a number of them, meritocracy may be considered one of the more common authority structures, or type of governance in OSS communities (e.g., [24, 58, 161, 166, 171, 193]). Based on merit and the earning of trust and status in the community, individuals are granted further responsibility and authority [58]. Merit correlates to the quality and quantity of the individual's contributions [74, 206]. A common assumption is that these contributions are limited to technical code contributions, however, as is shown by Eckhardt et al [51], this can be a simplification. Considering the onion model [161], several paths are depending on the type of role an individual possesses. Proven coordination and leadership skills are aspects that may be considered [101, 174], but not obviously captured in code commits. As highlighted by O'Mahony and Ferraro [174] in their study of the Debian community, *"Any examination of meritocracy must develop a context-specific understanding of how merit is conceptualized"*.

2.3 Influencing the Requirements Engineering Process in OSS communities

The members of the community all have their motives for participating, social or economic [124, 183]. It may, therefore, be considered a challenge for firms to align their internal agenda with that of the community [39, 173, 194]. A decision to add functionality may require consensus in the community and approval by the

community leadership depending on the type of governance. Being too aggressive with one's agenda may have an adverse effect and result in the functionality being blocked [2].

Dahlander et al. [38] differentiate how firms can adapt their relationship with an OSS community based on the level of influence needed. On a continuum scale, a relationship can be characterized as parasitic, commensalistic or symbiotic. In the parasitic approach, the firm takes without giving back, by some referred to as a "free-rider". In the commensalistic approach, the firm contributes back when motivated, but focus on internal development. In the Symbiotic approach, the firm also sees to the best of the community, working to align internal and external development. The alignment is created through working as peers, and building status and recognition inside the community [40].

To build a symbiotic relationship, firms should first understand and learn to respect the needs, norms, and structure of the community [2, 24, 38, 40, 136, 165], a form of "good citizenship" [173]. If there is a foundation encapsulating the OSS community, firms may have the option to gain influence through membership or sponsorship [171, 173], or in other ways supporting the foundation, e.g., by supporting development with infrastructure [38], or general subject matter expertise [24]. In return, they may receive seats at relevant boards and committees through which they can make their voice heard [24, 171]. Foundations, and similar boundary organizations between firms and an OSS community, are often limited to managing the technical direction of an OSS projects [173].

A more direct and general approach to the control of code contributions is by having "a man on the inside", letting employees engage with the community [40, 87, 165, 166, 173, 194]. An alternative is to contract members of the community directly to have them work on matters of importance to the firm [39, 74, 173, 185, 194]. Through their engagement, these sponsored community members can take part in the RE processes by participating in discussions and providing both technical and non-technical contributions and support [24, 157]. This work may take place both online and offline, because being visible and active on both ends is essential [157, 174, 194, 202].

2.4 Determining the need for Influence in OSS communities

As highlighted by Dahlander and Magnusson [39], it may be difficult to determine which OSS communities are of strategic importance to their operations. Firms should identify how they could benefit from an OSS project and its community, and then what kind of engagement is required to reap these potential benefits [24].

From a business model perspective, it may be considered how the OSS project helps to create, deliver, and capture value for a firm [207]. It may, for example, serve as a basis on which the firm builds complementary products or services, such as support and subscription offerings, or proprietary extensions [197]. The OSS

project could also function as a product or service enabler, embedded in hardware products [128], or as tooling and infrastructure for development and service delivery [157]. From a more strategic perspective, the OSS project may provide value as a foundation for pooled R&D/product development, and as a mean for standardization of technology [222]. Furthermore, just as the community may serve as an external workforce, it may also serve as a marketing channel, both for customers and future employees [39, 86, 185]. Hence, the value should be viewed both from a monetary and a non-monetary perspective [197].

From a technical perspective, it is also essential to understand the strategic connection of the OSS project to a firm's business and how this is reflected in a developer's level [24]. There may be internal dependencies and integrations between the OSS project and internal software that are critical to maintain [157], as is specific functionality that is requested and expected by the firm's customers [128]. These two reasons both warrant a need for alignment between software development inside the firm and the community respectively [39, 194]. If the direction of the community is predictable, both regarding road-map and release planning, then the need for an active community presence may be less urgent [24].

3 Research Design

To develop the CSF, we used a design science research approach [90, 228], in which research is performed and structured in the form of design cycles. A design cycle is comprised of three phases: problem investigation, artifact design, and artifact validation [228]. These phases are performed iteratively, as exemplified in Figure 2. For example, as artifact validation renders feedback, this feedback is used for refinements in artifact design, resulting in a new artifact design that needs validation. Below, we use this structure to describe how we planned and executed the research behind this study.

3.1 Problem Investigation Phase

In the problem investigation phase, the problem context is analyzed. In our case, we conducted exploratory interviews to understand industry practice beyond what has been identified in the literature.

Ten individuals were interviewed (denoted I1-10, see Table 1) with a semi-structured approach where the interview instrument consisted of open-ended questions (see Section 9, Appendix A). The interviewees all held positions with responsibilities relevant to understanding how their respective firms work and engage with OSS communities. They were selected based on convenience sampling. All interviews lasted between 30 to 60 minutes and were conducted either in person or over video link by the first author of this study. All interviews were audio recorded and transcribed.

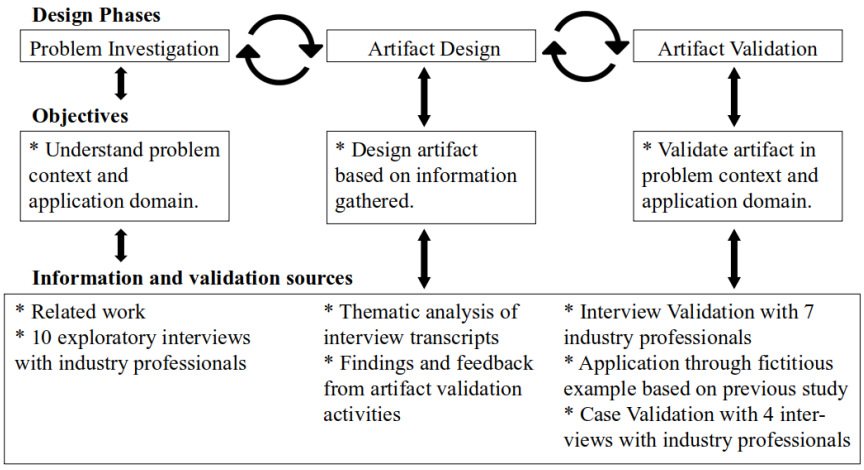


Figure 2: Overview of the research process and context used in this study, using design science research [90, 228]. Design steps includes problem investigation, artifact design and artifact validation, which are performed iteratively.

3.2 Artifact Design Phase

Drawing on the knowledge and understanding that is obtained during the problem investigation, an artifact is designed with the hypothesis that it will address the design problem. In this study, the design problem is stipulated by **RQ1** and **RQ2**, and the artifact is the CSF.

The interview transcripts were coded with an inductive approach by the first author with audit trails intact [191]. Sentences and paragraphs were first assigned descriptive topics. These topics were later collected under common codes, which could then be related and sorted under **RQ1** and **RQ2** respectively. Codes relating to **RQ1** are referred to as aspects and are divided into three categories; Business Aspects (BA), Technical Aspects (TA) and Community Aspects (CA). Codes relating to **RQ2** are referred to as engagement practices and are collected in one single category. The CSF is presented in full detail in Section 4

Below we provide an example with a subset of quotes rendering in engagement practice 5 (EP5) of the CSF:

- Engagement Practice (RQ2)
  - Offer the expertize and resources of the firm
    - Quote by I1: *" . . . contributing DevOps-kind of information and documentation and information, and it gives credebility"*.
    - Quote by I7: *"We don't have developers, but we send you these machines to do testing"*.

3.3 Artifact Validation Phase

In the artifact validation phase, the artifact is tested as a candidate solution to the defined design problem. In our study, this phase consisted of three steps. First, we conducted seven validation-focused interviews with four new industry professionals (I11-14), but also three from the problem investigation phase (I1, I5, I6), see Table 1. Second, to evaluate applicability and utility (i.e., descriptive validation [90]), the framework was applied through a fictitious example on a previously performed case study on how Sony Mobile evolved in their engagement with the communities of Jenkins and Gerrit [157]. Third, the CSF was validated in a similar way as in the first step, but within the context of a software-intensive firm (CaseOrg) and its Tools department.

Interview Validation

The CSF was presented and discussed one element (aspect or practice) at a time to the interviewees. Discussions focused on whether something was redundant, missing, or could potentially be modified. Interviews were audio recorded and transcribed.

After verifying with transcripts, this step of the validation phase resulted in one TA being removed, TA2 being reformulated to also focus on the "fitness-of-use", and the explicit addition of TA4. A further consideration brought up in several of the validation interviews was that, while the business and technical aspects are relevant for determining the need for influence on the RE process, the community aspects are on the other hand used for determining the feasibility and potential of gaining influence in the community. I13, for example, describes it as, *"So your first two sets of criteria, the business and technical aspects, felt like you were deciding yes or no, we should care about this community. The community aspects don't feel like yes/no's, we should care, these feel much more like feasibility, can we do it or not"*. Furthermore, the validation interviews resulted in the validation of and more nuances to existing aspects and practices. I12 for example added to BA2 the perspective that standardization can be part of a strategy to build a software ecosystem. From a design science perspective, findings and feedback from the validation phase were used to refine the artifact design.

Framework application example

The analysis was performed by the first author of this study, who was also one of the authors behind the previous case study [157]. Using interview transcripts and codings from the original study, the CSF was applied by considering each aspect against the Jenkins and Gerrit communities. Traces and support for the different engagement practices were then searched for. Findings were then summarized and verified for correctness with the OSS Program manager at Sony Mobile. The program manager was presented with the results from each of the applied practices,

Table 1: Ten industry professionals (I1-I10) were interviewed in the problem investigation phase. Seven industry professionals (I1, I5, I6, I11-I14) were interviewed in the Interview Validation phase. Four industry professionals (I15-18) were interviewed in the Case Validation. Small-sized firms (S): <50 employees, Medium-sized firms (M): 50 <>250 employees, Large-sized firms (L): >251 employees)

ID	Title	Firm	Business	Size	Use of OSS
I1	OSS Program Officer	A	Telecom	L	Infrastructure
I2	Community Manager	A	Telecom	L	Infrastructure
I3	OSS Program Officer	B	Software products	L	Infrastructure & Products
I4	OSS Strategist	C	Software products	L	Infrastructure & Products
I5	Community Manager	D	Software products	S	Products
I6	Community Manager	E	Software products	S	Products
I7	OSS Strategist	F	Software products	L	Products
I8	OSS Program Officer	G	Software products	L	Infrastructure & Products
I9	OSS Program Officer	H	Software products	L	Infrastructure & Products
I10	OSS Strategist	I	Consumer electronics	L	Products
I11	OSS Strategist	J	Consultancy	S	Strategy services
I12	Community Manager	F	Software products	L	OSS products
I13	Community Manager	F	Software products	L	Products
I14	OSS Program Officer	K	Consumer electronics	L	Products
I15	Team Manager	L	Embedded systems	L	Infrastructure
I16	Project Manager	L	Embedded systems	L	Infrastructure
I17	Senior Developer	L	Embedded systems	L	Infrastructure
I18	Junior Developer	L	Embedded systems	L	Infrastructure

and the support gathered for each of the engagement practices. The program manager was asked to verify the interpretation and clarify any misunderstandings of the first author's analysis. It should be noted that the program manager was also one of the interviewees from the previous case study [157].

Case validation

The Tools department has a similar organization and purpose as that described in earlier work of Sony Mobile, which is the foundation for the application example as described in Section 3.3. CaseOrgs's Tools department develops and maintains multiple OSS tools and infrastructure projects, including Jenkins and Gerrit, to support its product development organization. All OSS communities that were discussed during interviews were characterized as community-managed and meritocratic.

Four interviews were conducted with I15-I18 (see Table 1) who all held various positions but were all engaged in different OSS communities, some with maintainership positions. As in the previous step (see Section 3.3), the CSF was presented and discussed one element (aspect or practice) at a time to the interviewees. Discussions focused on whether something was redundant, missing, or could potentially be modified, specifically in the context of the communities that CaseOrg's Tools department is engaged in. Interviews were audio recorded and transcribed. Findings and feedback were used to refine the artifact design of CSF. No aspects or practices were removed or added. Existing ones were however given more nuances as EP5 where the importance of attending and arranging hackathons was added.

4 Community Strategy Framework

Here we describe the Community Strategy Framework (CSF) as presented in Table 2, which consists of two parts. The first of these, contain aspects a firm should consider when assessing its need to influence the RE process in an OSS community (**RQ1**). The second part of the CSF consists of practices a firm should consider to gain influence on the RE process in an OSS community with meritocratic governance or aspects thereof [198] (**RQ2**). Figure 3 shows an overview of the CSF. A firm constructs a community strategy by firstly assessing the community of interest based on four Business Aspects (BA1-4) and four Technical Aspects (TA1-4), and secondly determine the actual need for and feasibility of gaining influence using the four Community Aspects (CA1-4). It may be that not all aspects are applicable or relevant. It may also be that one aspect may indicate a need for influence, while another may not. With this in mind, it is up to the user to consider the different aspects in relation to the community of interest, and weigh these against each other. The CSF should, therefore, be viewed as a support for the user to arrive at a

decision on if and how much influence is needed by the firm on the RE process in the OSS community.

Once such a decision has been made, the firm then formulates important engagement goals and selects which Engagement Practices (EP1-8) to apply, and finally determine how to apply them. Below we present the respective aspects and engagement practices in detail.

For further guidance on how to apply the CSF, please see Section 5 where it is applied in a case example based on earlier work [157].

4.1 Aspects

The aspects are divided into three categories: business, technical, and community aspects. Aspects from the two former categories are used to reflect on the OSS and its importance to the firm from a business and technical perspective. The latter, community aspects, are used to reflect on the feasibility and potential to gain influence, as well as the need for it.

Business Aspects (BAs)

BA1 - Connection between the OSS project and the value proposition and revenue streams of the firm's business model. As expressed by I10, *"It comes down to the bottom-line, and making sure where [the firm] is making money, we want to have as much impact on those areas as possible"*. I11 emphasizes *"I think the sticky point is understanding how the value of the open source matches*

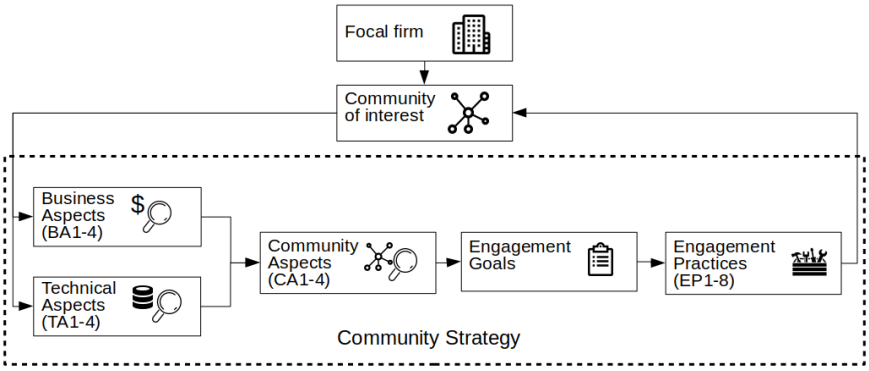


Figure 3: Overview of the Community Strategy Framework's related process. A firm first values the community of interest with the business and technical aspects and then uses the community aspects to determine the feasibility of gaining influence and potential engagement goals. Engagement goals are then decided and engagement practices chosen.

Table 2: Overview of the Community Strategy Framework. Business, Technical and Community Aspects relate to **RQ1** and Engagement Practices to **RQ2**

Business Aspects (BA)	
BA1	Connection between the OSS project and the value proposition and revenue streams of the firm's business model.
BA2	Connection between the OSS project and the business strategy of the firm.
BA3	Importance of the OSS community as a pool for recruitment
BA4	Need of the OSS community-related visibility and credibility towards the firm's customers
Technical Aspects (TA)	
TA1	Internal dependency of the OSS project inside the firm
TA2	Fitness-of-use and road-map alignment of the OSS project
TA3	Dependency on the OSS community's release planning
TA4	Need for competence and resources of the OSS community
Community Aspects (CA)	
CA1	Presence, influence and agenda of other stakeholders in the OSS community
CA2	Diversity and activity in the OSS community's stakeholder population
CA3	Openness in Culture and Governance of the OSS community
CA4	Ownership and management of the OSS project
Engagement Practices (EP)	
EP1	Understand the governance structure and have seats in right groups, committees and boards
EP2	Become a member or sponsor of the foundation or governing community body
EP3	Sponsor, contract or hire developers and maintainers to engineer contributions and mentor internal engineers
EP4	Contribute to the development of the OSS project through internal engineers
EP5	Offer the expertize and resources of the firm
EP6	Have an active on-line and off-line community presence
EP7	Be open and humble to the OSS community
EP8	Build an inner source culture and practice inside the firm

to the value of the business they're trying to build". A firm should, therefore, recognize how the OSS project is leveraged in its business model. It can be a complement of the core value proposition as for Red Hat and their distribution Red Hat Enterprise Linux which is based on Fedora. It could also be an enabler for the value proposition as for Sony Mobile and Android which is used in their mobile handsets. Or it could play a more indirect role as part of an infrastructure or a tool-chain that can be used to develop and deliver the main value proposition. In the latter case, there may be a limited amount of competitive edge connected to how the OSS project is used internally, as described by I16, *"We're so far out of core business that we have our own contribution process"*. In other cases, *"if you're building a product offering around an open source project in the core, there's not even a question. If you're committing to customers to support the project, then you need to have an influence on that project"* (I12).

BA2 - Connection between the OSS project and the business strategy of the firm. The business strategy specifies how a firm should navigate a changing environment and as a consequence construct and adapt its business model [45]. In this context, an OSS project and its community can play a pivotal part, e.g., to commoditize a market, or change the default technology being used by industry. I9 reflected on one of their experiences, *"So we wanted to change the industry conversation, and we wanted to have a substantial impact in that"*. This type of standardization can further be part of a strategy where the intent is to build a software ecosystem, as explained by I12, *"Driving standardization enables the market to potentially develop and that is what gives business opportunity, if you're running an infrastructure project and all of a sudden you have a lot of third-party vendors, whether monitoring and logging, or storage or network, you know there's an entire ecosystem that comes along, not to mention all of the developer toolings that you need to develop container-native applications. So there's a lot of opportunities that come from having a de facto technology base in the platform. And that creates that opportunity to create the commercial ecosystem around the platform"*.

BA3 - Importance of the OSS community as a pool for recruitment. Being active and influential in an OSS community can be important to attract and maintain a skilled workforce. This concerns both specific technologies where skilled people are scarce and attracting developers in general. The latter is emphasized by I7, *"It's also about reputation - [firm] created a big open source office, and a huge open source initiative, because no one wanted to work with them"*. I11 adds, *"This is something a lot more are starting to realize, particularly large companies with aging populations, that people don't want to sit in a stodgy old company in cubicles"*. I16 continues, *"We need to show that we don't just consume, but also contribute to attracting good developers. The community becomes a channel for new employees"*.

BA4 - Need of the OSS community-related visibility and credibility towards the firm's customers. As for recruiting talent, being active and influential in an OSS community can be essential to attract and maintain customers. It can be to prove technical competence, but also the ability to push features upstream. As put by I1, *"[The OSS project] was the selling point of the product. We needed to demonstrate to customers that we were one of the core contributors of [the OSS project]"*. I11 gives the example of IBM and how they, *" ... back in the 2000s, invested a billion dollars in Linux and they wanted to make a big deal of it because they saw that as an emerging market that they wanted to get into"*. I1 adds how this aspect is particularly important for firms using OSS in their products, such as *"Red Hat, or any commercial open source project. Like Cloudera would need to do that, that they have influence in the Hadoop community, and DataStack for Cassandra"*.

Technical Aspects (TAs)

TA1 - Internal dependency of the OSS project inside the firm. Technical dependencies between an OSS project and a firm's internal software can be considered an architectural reflection of how an OSS project connects to a firm's value proposition (BA1). Certain features in a product or parts in an infrastructure may be dependent on the project. I3 phrases the question as *"Do you depend on this or do you not? And how much do you depend on it? How much functionality goes into your product that's based on upstream software? How heavily are these integrated into your product?"*. I17 exemplifies, *"We are extremely dependent on [OSS project], we have based our whole infrastructure chain on it. This requires us to be active so that we can affect in what directions the tools head"*.

TA2 - Fitness-of-use and road-map alignment of the OSS project. Deviance between a firm's internal and a community's requirements and road-maps may be essential to address in order to avoid or minimize technical debt. I12 refers to an OSS project's fitness-of-use and explains it as *"How many things that we need it to do does this project do today? And if you feel like there is a delta between what it does today, and what you need it to do, and this is a strategically important component of your plan, then it would be important to be involved. You need to have influence so that you can affect the change that you need in that project"*. I1 adds that in *" ... some cases, it may not be necessary for you to be as actively involved because you are happy with the direction it is going, and it's a mature and stable project. And in some cases, you really need to be there and watch it and make sure it goes in the right direction"*.

TA3 - Dependency on the OSS community's release planning. A firm can be more or less dependent on the release planning of an OSS community, and have various needs to synchronize it with that of any internal development. Getting features upstream quickly and running the latest release may be an essential factor for firms whose customers may expect quick access to the latest functionality, as some buyers do of Android-based mobile handset manufacturers. For others, it may be less of a concern, as for Red Hat who focuses on offering a stable and secure version of Fedora. I4 explains it as *"How much do we care if they are changing it rapidly? Are we living on a fork and are willing to eat a little bit of fit and finish? Or do we really want to be on the latest bits all the time?"*. I12 sees it from a risk analysis perspective, *"Is there is a risk that a feature will not go into a project, or is there a risk that a project that you depend on will miss its release date?"*.

TA4 - Need for competence and resources of the OSS community. Firms can be limited, both in terms of *" ... resources, time or people"* as highlighted by I16, or in terms of specific competencies that are internally available. By engaging in an OSS community, firms may have an opportunity to gain these resources through collaboration and "co-opetition". By growing influence in such a community, a firm can better exploit and steer these resources to best match the firm's

agenda. I1 explains it as, *"Sometimes we may not have the competency inside the company, but yet we want to draw on the competency of the community to help us use something correctly. So, the link to the community may be important because of that, to just improve our own competency in handling that project"*.

Community Aspects (CAs)

CA1 - Presence, influence, and agenda of other stakeholders in the OSS community. Knowing whom the stakeholders are, where they focus their resources, with whom they collaborate and how much influence they hold, can signal how a firm should consider its relationship with the stakeholder, but also overall community engagement. As expressed by I3, *"If it's a company that wields a big influence, and they are a competitor to you, there's a much different way to approach that than if they were a partner or one you're not a competitor with"*. I3 continues, *"If you understand why those companies are contributing it potentially makes your strategy why you should be contributing"*. I1 explains that for projects which are important to a firm, *"You work on it, you contribute to it, and you make sure your competitors are not influencing it differently than you would"*. Hence, the presence of competitors may indicate *"how strongly [a firm] need to be present"*, as further highlighted by I1. However, as explained by I13, OSS communities provide a *" ...forum for competitors to cooperate in a way that doesn't upset their shareholders, a form of co-opetition"*. Presence of competitors may also be *"a signal that the OSS project we should be engaged in, maybe it is becoming an industry standard"*, as suggested by I2.

CA2 - Diversity and activity in the OSS community's stakeholder population. A community maintained by only a few individuals or companies could be vulnerable if they were to leave for any reason. As asked by I3, *"What would be the case if there were shortages in supply, i.e., the project would no longer be available?"*. I12 compares a community to an external vendor and asks, *"Is this company going to be in business in five years? Is there someone who can take up the mantle if they shut down?"*. A low level of diversity and activity in a community can, therefore, be a warning sign if a firm is to engage in the first place. However, if a firm is dependent on a community or sees potential, then it indicates that the firm should invest, *" ...not for influence, but for health"* as emphasized by I1. I9 further adds, *"We want to make sure it is not just totally dependent on one or two parties only because vibrant community to me means that it has a broad spectrum of contributions and that it is not just totally dependent on one party"*. From a sourcing perspective, it may be relevant to also consider other alternatives and weigh these against the cost of investing in the concerned community.

CA3 - Openness in Culture and Governance of the OSS community. To be attractive, the culture and governance of an OSS community should have meritocratic influences, i.e., be open to new members joining and gaining in rank, but also for discussions regarding road-maps and ways of working to be open. This is

further explained by I9, *'If there are communities that are uninterested in changing and learning then that is a community in my opinion that will stagnate and contract, they will have a hard time growing new leadership, they will have a hard time evolving as the needs of users evolve, as the needs of community evolve"*. I9 continues, *'An openness to constant improvement and to input needs to be a core value, or at least be demonstrated in a community in order to consider putting any substantial investment"*. If the project is important for the firm, a low level of "openness" could motivate a high investment and active engagement to be able to affect the culture and governance of the community if deemed possible.

CA4 - Ownership and management of the OSS project. A criterion before engaging in an OSS community is to determine whether there is potential for the firm to gain influence and extract the expected value. As highlighted by I10, *"If we see that it's a project that is controlled by one company, and it doesn't look like we'll be able to influence it in a way we want, we may not get involved in that project"*. I.e., if the OSS project is now owned and managed by the community, or a legal entity representing it (e.g., a foundation), gaining influence through active contributions and engagement may prove hard. If a firm's strategy is to hire a maintainer to get influence and there is no one available, *" . . . the project becomes much less attractive"*, as stated by I10.

4.2 Engagement Practices (EPs)

The engagement practices presented in this section should be seen as a tool-box of ways in how a firm can engage with a meritocratic community to build the influence needed.

EP1 - Understand the governance structure and have seats in right groups, committees, and boards. Depending on the complexity of the community governance structure, there can be many groups and committees where decisions are made. As described by I13, *"It's very dependent on the community, some have large foundations, while others may have less"*. Hence, a firm should first *". . . understand where decisions get made and what kinds of decisions [they] need to influence. Is it a technical decision? Is it a positioning decision? Is it a communication decision? And hence, which body do you need to be on, or what level of membership do you need?"* as stated by I1. Once understanding the governance structure of the community, a firm may need to build a certain level of influence to be able to join the identified groups. This need can also concern groups and committees that may not be a direct part of the community, but part of a greater ecosystem affecting the OSS community. As expressed by I1, *"It's an influence game making sure you have people in all the right places, joined all the right foundations"*. I2 provides an example, *"I think with [OSS community] they did that, ok, we need to sit at this group, this group, this group, we need to get a seat at the table of the user committee, we need to be on this committee, and they actually mapped it, and they put people there"*.

EP2 - Become a member or sponsor of the foundation or governing community body. Once a firm understands the community and its governance structure, they can start to consider whether they should become a member or sponsor if possible. If the community is run under a foundation, a membership can give a firm visibility and marketing to show community, customers, and potential newly-hires that they are both competent and committed in regards to an OSS community. However, it does not have to imply a direct influence on the OSS community automatically. As explained by I3, *"You can potentially buy yourself into the business side of the governance, but you don't get any technical influence unless you do any work"*. I13 gives the example of GNOME, *"You pay to be part of the advisory board, but it has very little power. You have to be a contributing member to be elected to the board of directors, and that's where the power is"*. A membership or sponsorship should instead be seen as a long-term investment that can help build a sustainable influence through growing and attracting influential community members and maintainers. Sometimes membership may be unnecessary, as explained by I1, *"A lot of these bodies have end-user boards which do not require any pay-to-play, it just requires you to be a big user of that technology. Because a lot of projects are very eager to get feedback on how you are using it, what are the challenges that you face at scale? So they see that as currency and value. So we've kind of been reexamining our presence in some of these bodies and asking why are we spending 40K when we can get the same influence through being on the end-user committee?"*.

EP3 - Sponsor, contract or hire developers and maintainers to engineer contributions and mentor internal engineers. To build influence organically by on-ramping new developers into a community can be time-consuming, why a firm may consider hiring existing maintainers and developers in leadership positions. As explained by I3, *"If your willing to do a longer-term play, then you get people already in your development team starting to make upstream contributions, then it may take a year or two years depending on what kind of community it is, to have the influence long term. But if you needed that influence yesterday, the only way is to hire someone that is a very strong contributor or maintainer"*. I10 adds, *"We like to hire people in leadership positions. And once we get to two or three people that are in those sort of positions, then we can get started introducing some junior developers"*. This kind of mentoring is further endorsed by I12, *"I would hire the contractor to teach how to do the work. So it's kind of on-the-job-training"*

EP4 - Contribute to the development of the OSS project through internal engineers. Long-term and sustainable influence is built by directly contributing to the development of the OSS project. These contributions are not limited to code, but may also include *" ... writing documentation, testing, answering questions, doing the mud work, doing a lot of the things that no one wants to do"*, as explained by I3. Developers need to be enabled to actively engage in the community development process without being hindered by internal contribution processes. I17 adds, *"Principally all open source communities are run as meritocracies so we*

need to be active. If we want to be able to change the direction in [OSS project], we need to produce code and plugins to show that we are part of the community".

EP5 - Offer the expertize and resources of the firm. If a firm holds specific resources, these can also provide valuable contributions to the community. These resources can, for example, be infrastructure-related, but also include soft factors. I1 exemplifies how they provide large-scale testing capabilities, as well as credibility to an OSS project that they run in production, *"if you have big companies like us using [OSS project], it says that it is a viable product"*. I9 adds another example where they provide server space for the community to run compute, test and build processes, enabling the active development in the OSS community.

EP6 - Have an active on-line and off-line community presence. Community discussions regarding the development of an OSS project take place in on-line mediums such as issue-trackers, chats and social media, but also off-line at events and social gatherings such as meetups, conferences, and hackathons. For a firm to grow and leverage its influence, it needs to be present and take an active part in these discussions, and also help to facilitate them, e.g., by arranging their own events. I17 exemplifies, *"Concerning [OSS project], we are extremely active at hackathons... We travel a lot to get and know the people... Recently we hosted a hackathon where we gathered basically all maintainers of the project"*.

These activities should be coordinated internally as highlighted by I1, *"[The Community Manager] ran community activities internally and externally, created awareness of what we were doing in [OSS community], making sure that people contributed to the right projects, submitted abstracts to the right projects, were elected to the right bodies, showed up at the right conferences"*. Having dedicated developer advocates and community managers was a generally recommended practice. This person should be able to mediate and be a spokesperson both of the community and the firm.

EP7 - Be open and humble to the OSS community. When joining a community, I9 explains, a firm should adapt to the culture and way of working in the community. They should *" ... come in humbly and offer to help in things where they have expertize as opposed to 'We need to do a thing, it's gotta be done this way', then you are going to get an immune reaction if you start that way"*. I12 gives the comparison, *"Joining a new open source community is like moving into a new neighborhood. There is a way of doing things, some of these things are going to be built up during time, and there is going to be inertia. So there are things that are obviously better, that people are going to agree is obviously better, but they are used to the way things are done. So you kind of have to figure out how to bring change gradually. And at the same time is that you figure out how things work. And so, first figure out how the community works before you come in and propose a lot of changes, so don't be excessively critical of the way things are done in a specific neighborhood, then no one is going to listen to you when you propose changes"*. Furthermore, the firm should be *" ... transparent and open about the intentions and the agenda with the community and project, e.g., road-map, what*

you are keeping closed and for what reason" as highlighted by I7. This is further emphasized by I13, *"You need to be open and completely honest about the why even if it's for profit because otherwise you just look suspicious"*. The firm should hence differentiate between the communication they use towards the community and that which they use towards for example their employees and customers.

EP8 - Build an inner source culture and practice inside the firm. By introducing inner source culture and development practices internally, a firm can help its developers to learn better how to work with external OSS communities, and simplify on-ramps. I13 explains the importance of teaching internal engineers about OSS development practices, such as working distributed and decentralized, *"A lot of companies where everyone sits in the same office all talking among themselves and have meetings just them, and they need to learn how to do everything online, and include the people who are not there"*. In effect, this can create more contributors for the firm and in a longer perspective help raise its influence in OSS communities in general.

5 Framework application example: Jenkins and Gerrit

In this section, we illustrate through a fictitious example of how the CSF could be applied (cf. descriptive validation [90]), based on a previously studied case which describes how Sony Mobile and its Tools department evolved in their engagement with the communities of the two OSS projects, Jenkins, and Gerrit [157].

Initially, Sony Mobile had a restrictive view of what they shared with the two communities and how they engaged. They focused mainly on doing bug-fixes, general knowledge-sharing and had a community presence limited to online channels, such as mailing lists and issue trackers. The engineers in the Tools department focused on internal work and tailoring of the two OSS projects to internal needs. They further saw that they could create a competitive advantage by keeping internally developed features closed. However, with time, the attitude towards the two communities evolved into a more symbiotic relationship. Sony Mobile and the engineers at the Tools department saw increased benefits with having an active engagement and being more open. As then highlighted by Sony Mobile's Director of OSS operations (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"*. It is in this context that the aspects of the CSF are analyzed and discussed for Jenkins and Gerrit.

5.1 Defining the need for influence in the Jenkins and Gerrit communities

Below we investigate the need for influence in the Jenkins and Gerrit communities by considering the business, technical and community aspects of the CSF from Sony Mobile's point of view.

Business Aspects

The connection between the two OSS projects and the business model of Sony Mobile (**BA1**) was indirect in the sense that the projects were used in the development infrastructure that engineers leverage in the product development inside Sony Mobile. Perceived benefits from the use of the two OSS projects include improved quality of Sony Mobile's end-products, as well as shorter time-to-release and market. Both OSS projects were seen as a commodity and a non-competitive advantage. There were alternative solutions available, but most were proprietary, and the primary motivation for an OSS option was that Sony Mobile could customize the OSS projects based on internal needs much more easily.

The adoption of Jenkins and Gerrit was part of a broader strategy (**BA2**) of moving Sony Mobile more towards usage of OSS, as well as the adoption of the same tool-chain used by Google in the Android development.

Both communities made up important pools for finding and attracting new and talented employees that could help in adapting the two OSS projects to the preference of Sony Mobile (**B3**). However, as neither Jenkins or Gerrit was a part of the product or any marketing, there was no need to establish a certain level of visibility or credibility towards the customers (**B4**).

Technical Aspects

Both Jenkins and Gerrit made up pivotal parts of the continuous integration tool-chain inside Sony Mobile. Therefore, there were many interactions and dependencies between the two OSS projects and as well as to other tools in the tool-chain (**TA1**). They had been tailored to internal requirements and supported the development process defined internally.

Sony Mobile was dependent on a stable and secure infrastructure, why they did not need to use the latest or experimental releases (**TA3**). In general, however, there was an expressed goal to avoid too many patches, and adaptations as the Tools department was limited in resources and had to rely on the community for much of the development (**TA4**). Further, there was a need to introduce a heavier focus on scalability in the two OSS projects, as they at the time were not optimal in large-scale setups as that used by Sony Mobile (**TA2**).

Community Aspects

Both the Jenkins and Gerrit communities had several firms involved, including direct competitors to Sony Mobile. However, as both OSS projects were seen as non-competitive by Sony Mobile, this presence was not considered as an issue. Few of the existing stakeholder had the equivalent or larger size of installations, which made Sony mobile somewhat unique in its need for improved scalability (**CA1**).

In general, both communities were very active and diverse concerning contributors and users (**CA2**). Also, the culture and governance structure was very open for new contributors to join in discussions and rise in rank (**CA2**). Due to the healthy activity, and meritocratic culture and governance of the communities, it was also deemed easy to increase influence, both organically by introducing employees, but also through hiring new talent as both communities are community-managed(**CA4**).

Summary and Goals for Engagement

Even though classified as non-competitive, there was a defined need to be able to influence the road-maps of the OSS projects, and be able to contribute larger features (e.g., related to improving scalability). Due to the limited size of the Tools department, there was also an expressed goal to be able to find and create collaborations when possible, even with competitors.

5.2 Defining the engagement activities in the Jenkins and Gerrit communities

Based on the determined need for influence and goals that were defined, Sony Mobile and its Tools department became more active and open in their engagement with the two communities.

No foundations were surrounding the two projects, why there was no need to attain a specific membership or sponsorship (**EP1**). However, there were committer groups (**EP2**), i.e., central parts in the communities' governance [161], that Sony Mobile wanted to join.

Sony Mobile did not see a need to rush and hire engineers from the communities directly (**EP3**). Instead, they grew their influence organically by introducing their engineers to the communities to the point where they managed to get to positions in the Gerrit committer group. The engineers were given frame agreements for the two communities where they were allowed to contribute freely, both in regards to features and bug-fixes (**EP4**). With their large set-up and testing infrastructure, Sony Mobile could also contribute to improving the quality of the two tools (**EP5**).

The engineers at the Tools department were active and visible in both online and offline communication channels (**EP6**). Their online presence included ac-

tive participation in discussion and knowledge sharing through mailing lists, issue trackers, chat channels, and webinars. Offline presence included attending conferences, meet-ups, and hackathons. The latter was seen as an essential forum to do quick implementations (cf. just-in-time RE [54]) as often many of the more influential persons in the communities were gathered in the same room. Alongside this active engagement, Sony Mobile had an open attitude towards the community and was transparent with its agenda. Engineers presented how the two projects were setup internally, as well as best practices and know problems when possible. They even talked to and engaged in knowledge-sharing with direct competitors (**EP7**).

The engagement and internal development of Jenkins and Gerrit were further seen as a seed to create an inner source initiative inside Sony Mobile, with the ambition to spread into others corners of the Sony Corporation (**EP8**). The goal was to grow more contributors and active users to Jenkins and Gerrit, but also in other projects, and maybe even create new ones where motivated.

6 Discussion

Below we discuss the validity of the CSF and contrast it to related work.

6.1 Determining the need for Influence in OSS communities

From a business perspective, as highlighted by I11, the *" ... sticky point is understanding how the value of the open source matches to the value of the business [a firm is] trying to build"*. In this sense, the business model concept provides a useful lens to frame how the OSS helps to create, deliver, and capture value for a firm [207], and more specifically, through the OSS project's connection to the value proposition and revenue streams as pointed out in BA1 of the CSF. As indicated by the diversity of the firms that the interviewees represent, this connection can be made in several ways, as is reported in literature [32, 157, 183, 197]. This is also true on the business strategy level where the firm chooses and configures its business model to compete in its business environment [45]. Creating or supporting a competing standard, or commoditizing a technology or market are two ways in how a firm can disrupt their competition and pave the way for their own business model, both reported on in the CSF (BA2) and in literature [222]. This alignment between the CSF and literature is further repeated in regards to the importance of an OSS community as a pool for recruitment (BA3), as well as a marketing tool towards customers (BA4) [39, 86].

On an implementation level, it is also important to understand the reflections between how the OSS project is used in the internal development and it's strategic importance to a firm [24, 194]. In the example of Sony Mobile and the communities of Jenkins and Gerrit (see Section 5 and [157]), the two OSS projects played a

less direct part in the firm's value proposition, but a much more significant from a technical perspective. They constituted core parts in the internal development infrastructure (TA1), and the communities were key partners to adapt and maintain the software (TA4). As the fitness-of-use and road-map alignment were not satisfactory, a high level of influence was required (TA2). If the case was otherwise, and the direction being predictable, the need for an active community presence may be less urgent [24].

Supportive evidence and alignment can hence be found between literature and many of the aspects identified in the interviews and presented in the CSF. However, aspects that need consideration may be different depending on the firm and community. For example, Sony Mobile had in regards to their customers, no need to prove credibility or visibility in the Jenkins and Gerrit communities, as these OSS projects were used internally and not part of any marketing or key selling points [157]. Hence, the business aspect BA4 is not relevant in this case, while in other cases it may.

Other aspects though can be considered more general such as business aspect BA1. An OSS project can, depending on the case, have a more direct or indirect connection with the value proposition and revenue streams of a firms business model. For Red Hat, the connection may most often be direct as they base many of their products on them [32]. Conversely, returning to the example of Sony Mobile, Jenkins and Gerrit had a more indirect connection as they enabled a customized development process. As reported, Sony Mobile experienced these community engagements as having a positive impact on time-to-market and quality of their products [157].

6.2 Influencing the Requirements Engineering Process in OSS communities

As reported in the literature (see Section 2), the type of governance in OSS communities can vary [26, 46, 47, 198]. There is also variation in the possibilities and ways how firms can gain influence on the RE processes in a community.

Among the cases researched, meritocracy seems to be among the more common authority structures [24, 58, 161, 171, 193]. In a meritocratic community, influence on the RE process is gained by proving merit, and as highlighted in the literature, this does not have to be limited to technical contributions [51, 174]. In essence, it is about earning the trust and status among one's peers in a community [40, 165, 166]. Considering the differentiation by De Noni et al. [47] between open-source based or collective communities, this characteristic can be assigned to both, even though the former is described as institutionalized and democratic, and the later as collective and meritocratic. In a purely democratic community, an individual still needs to earn trust, respect, and recognition among its peers to gain responsibilities and authority. In Apache communities, for example, both democratic and meritocratic traits can be found as individuals are voted into leadership posi-

tions even though Apache communities are profiled mainly as meritocracies [58]. Further, as De Noni et al. classifies Apache communities as open-source based, rather than collective, one can view the two categories of authority structures as closely related, as is the way in how influence can be gained on their communities' RE processes.

In communities with a centralized and autocratic authority structure [46], i.e., firm-sponsored or tolerant dictator-based [47], project leadership is often centered to a single (or limited number of) firm(s) (e.g., Android Open Source Project) or person(s) (e.g., the Linux kernel project). In firm-sponsored communities [26, 172], specifically those centered around a single firm [186, 194], where the focus is more on transparency than accessibility [224], communities may often be viewed more as user communities and a less open type of software ecosystem [71, 98]. To gain influence in these types of communities, firms may focus more on direct business relationships (cf. [10, 211]). However, this does not prevent meritocratic governance aspects to be present why the community engagement practices as proposed by the CSF may still be relevant. In tolerant dictator-based communities, as shown by Shaikh and Henfridsson [198], there can still be mixes of meritocracy and democracy implemented through different coordination processes. Even if such coordination practices would not be present in an autocratic community, there is still some possibility to influence by earning trust and respect in the community. If a firm can create enough traction among their peers in the community, the project leadership will commonly consider it [74, 120, 157, 167]. If not, and an opposing will is strong, part of the community may in worst case move to create their own fork of the project [168], as was the case with OpenOffice and LibreOffice [66].

Findings from the interviews regarding engagement practices align with Dahlander and Magnusson [38], in that influence in a meritocratic OSS community is built through creating a symbiotic relationship with the community. Trust and status are gained through active involvement and respecting its norms and values [2, 38, 166, 206]. As pointed out by S9, gaining influence may be done through different types of engagements and with varying types of resources, *"It's bringing code, bringing people into influence, in most projects, buying influence is not as easy to do, but you can still spend sponsorship money and port money to make sure that a project is happier or healthier for example"*. I.e., influence may, for example, be gained through providing code contributions as well as more general resources, including financial, aligning with practices reported in literature [24, 40, 87, 157, 165, 166, 173, 174, 202]. When comparing the communities inside the Linux Foundation and the Apache Software Foundation, S9 describes it as, *" . . . you need to show contribution, activity, commitment, leadership, and then you grow through contributions that you take in both foundations"*. Hence, the engagement practices in the CSF are primarily intended for OSS communities where there is a presence of meritocratic coordination processes [198].

7 Threats to Validity

As presented in Section 4, the CSF covers a broad spectrum of aspects, some more general and applicable than others. One reason for this may be that the 18 interviewees each have extensive personal experience in the field but with different backgrounds, e.g., business or developer-oriented. Another reason may be that they represent 12 different firms (see Table 1) which in turn may have different use cases and needs. From an external validity perspective [191], this is positive and indicates a potential of transferability to other firms who are engaged (or are aspiring to) in meritocratic OSS communities. However, as the CSF is based on qualitative data from a limited set of interviewees, quantitative conclusions on generalization will require further validation using statistics based on a population of real-world firms and OSS communities.

Another area regarding external validity is to what extent the practices presented by the CSF actually leads to a gain in influence, and in what contexts. As discussed in Section 6.2, we believe that there has to be meritocratic coordination processes present [198] as the engagement practices proposed in the CSF present make up different ways in how a firm can contribute to and engage with a community to build a symbiotic relationship based on trust, respect, and recognition among its peers. Interviews from the case validation (see Section 3.3) supports these arguments as the communities that CaseOrg is engaged in, and in the context which CSF was discussed, were all community-managed and meritocratic. However, external validity is still a limitation in regards to CSF why further empirical validation is needed in future studies, e.g., through the use of case studies and cross-case synthesis.

Regarding the completeness on the aspects and practices presented in the CSF, we again acknowledge that CSF is based on qualitative data from a limited set of interviewees. When performing the interviews in the Interview Validation step (see Section 3.3) of the validation phase we did reach a point of saturation where we observed a tendency of maturity in terms of declining number of emerged codes. This observation was further supported in the Case Validation (see Section 3.3) as no aspects or practices were added or removed, only further refined. This may point to some level of completeness. However, as presented in Section 2, there are numerous variations in the characteristics of OSS communities, e.g., in regards to governance structure, demographics, and RE process. Hence, further research and design cycles are needed to validate the CSF and to improve its level of completeness.

8 Conclusions

The focus of this study has been to identify what aspects that firms should consider when they assess their need of influencing the RE process in a meritocratic OSS community (**RQ1**), as well as what practices that should be considered in order to gain this influence (**RQ2**). To address these questions we used a design science approach [90, 228]. We developed a questionnaire used in ten semi-structured interviews with industry professionals. Inductive coding of interview transcripts [191], an initial version of a Contribution Strategy Framework was developed. The framework was then validated and refined through seven new interviews and by applying it on a fictitious example of an earlier reported study [157]. Finally, a case validation was performed by interviewing four industry professionals from a software-intensive firm engaged in multiple OSS communities. Questions focused on the validity of CSF in the context of the firm's community engagements. In total, 21 interviews were conducted with 18 industry professionals from 12 different software-intensive firms.

The framework consists of aspects and engagement practices. The aspects address **RQ1** and are divided into business, technical, and community aspects. The two former may be considered to help determine how important an OSS project and its community is from the business and technical perspectives, while the community aspects add the perspective of feasibility and potential in gaining influence in a community. The engagement practices address **RQ2** and should be seen as a tool-box of ways in how a firm can engage with a community to build influence needed in the community.

As this study uses a qualitative survey approach with a limited sampling of interviewees, further research is needed to validate the CSF through case studies and additional empirical work, both qualitative and quantitative. Along with such research, more theory-grounding work should be performed to further formalize the concept of influence in OSS communities, and how it can be gained, as exemplified by the CSF. Inspiration may be gathered from Valença and Alves [211] in how they generated a theory of power for emerging software ecosystems formed by small-to-medium sized firms.

Acknowledgments

The authors would like to thank the anonymous interviewees for lending their time and expertise, as well as the anonymous reviewers for their valuable feedback. This work was funded by the Swedish National Science Foundation Framework Grant for Strategic Research in Information and Communication Technology, project Synergies (Synthesis of a Software Engineering Framework for Open Innovation through Empirical Research), grant 621-2012-5354, and the industrial excellence center EASE (Embedded Applications Software Engineering)[2].

[2] http://ease.cs.lth.se

Appendix A - Interview questionnaire

- Do you, in any way, consider or plan how you engage with a community, and where you spend your resource, and to what extent? If yes, how? Is it formalized in any way? How could this be improved/otherwise done
- In what ways can you contribute to an OSS community (code, knowledge, socializing, sponsorship)? What roles would you say are involved in these contributions?
- How can you gain the power to change or affect (influence) a community in terms of what features gets implemented, and how they are prioritized? (Short- and long-term)
- Do you see any connection or consideration between how you engage and invest in a community and the level of influence you need to have in it? How would you describe it?
- How can you consider how an OSS and its community creates value for your company? Is there any relation or consequence between this value and how you engage and invest in the community, and the influence you need in it?
- Are you aware of the product planning and development in the OSS communities you are involved in? Are your internal product planning and development aligned with the OSS communities'? Do you consider possible dependencies? Do you see a need for it? How would it affect how you engage and invest in the community and the influence you need in it?
- Do you consider the motivation and underlying drivers for why you engage and spend your resources in a community? If yes, how? Do these align with how the community creates value for you and its role in your product strategy? Do they align with what you contribute to the communities? What do you see as the main drivers of your company?

CHAPTER IV

A METHOD FOR ANALYZING STAKEHOLDERS' INFLUENCE ON AN OPEN SOURCE SOFTWARE ECOSYSTEM'S REQUIREMENTS ENGINEERING PROCESS

Johan Linåker, Björn Regnell and Daniela Damian.

Abstract

Background: For a firm in an Open Source Software (OSS) ecosystem, the Requirements Engineering (RE) process is rather multifaceted. Apart from its typical RE process, there is a competing process, *external* to the firm and inherent to the firm's ecosystem. When trying to impose an agenda in competition with other firms', and aiming to align internal product planning with the ecosystem's RE process, firms need to consider who and how influential the other stakeholders are, and what their agendas are. **Aim:** The aim of the presented research is to help firms identify and analyze stakeholders in OSS ecosystems, in terms of their influence and interactions, to create awareness of their agendas, their collaborators, and how they invest their resources. **Method:** To arrive at a solution artifact we applied a design science research approach where we base artifact design on literature and earlier work. **Results:** A Stakeholder Influence Analysis (SIA) method is proposed and demonstrated in terms of applicability and utility through a case study on the Apache Hadoop OSS ecosystem. SIA uses social network constructs to measure the stakeholders' influence and interactions and considers the special

characteristics of OSS RE to help firms structure their stakeholder analysis processes in relation to an OSS ecosystem. **Conclusions:** SIA adds a strategic aspect to the stakeholder analysis process by addressing the concepts of influence and interactions, which are important to consider while acting in collaborative and meritocratic RE cultures of OSS ecosystems.

1 Introduction

Firms that use Open Source Software (OSS), e.g., as part of their supporting infrastructure, product strategy or business model, need to consider the Requirements Engineering process of the OSS itself [159]. This second, *external* to the focal firm, RE process is facilitated by the software ecosystem (cf. OSS community [161]) that surrounds the OSS [97]. Firms that are users of the OSS may also be involved in its development and maintenance and can be considered as members of the ecosystem, as well as stakeholders to the OSS. We refer to Glinz & Wieringa's definition of a stakeholder as *"... a person or organization who influences a system's requirements or who is impacted by that system"* [77]. In our context, we consider a person or an organization as the members of an OSS ecosystem, and the system being the OSS that underpins the ecosystem, using the definition by Jansen et al [97].

RE practices in OSS ecosystem may be described as informal and decentralized. There is often 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 [4]. Instead, RE may be considered as a lightweight and evolutionary process of requirements refinement [54]. Practices such as elicitation, specification, and prioritization overlap and are done collaboratively through iterative and transparent discussions and negotiations including up-front implementations [54, 74, 192]. These discussions and implementations of requirements are spread out over a number of requirements artifacts, each with its own repository. Examples of these artifacts (cf. informalisms [192]) include reports in an issue tracker, messages in a mailing list, or commits in a version control system. Prioritization is commonly conducted by stakeholders with central positions in the ecosystem's governance structure [10, 120]. To gain such a position in OSS ecosystems with a meritocratic governance structure, a stakeholder needs to prove merit by being active, contributing back, and having a symbiotic relationship with the OSS ecosystem [38].

Hence, the focal firm is one stakeholder among many within an open and fluctuating population in the OSS ecosystem [101]. This can result in conflicting agendas and lack of control, e.g., in regards to which requirements to be implemented and prioritized, render misalignment with internal RE processes [231], and complicate contribution strategies [159]. The focal firm may, therefore, have to gain

the influence necessary to affect the RE process in an OSS ecosystem according to its own agenda.

The Merriam-Webster dictionary [1] defines influence as *"the power to change or affect someone or something"*. In our context, this relates 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 [77], and as a characteristic enables firms to, e.g., see the requirements in which stakeholders hold a certain interest, and from there be able to create an overview of their agendas in the ecosystem [65]. Further, this understanding enables the focal firm to analyze how these stakeholders invest their resources in order to satisfy their agendas [65]. By also considering other stakeholders' interactions within the ecosystem, firms may identify possible partners and competitors [189]. Moreover, this can help firms to learn how to adapt their own strategies and processes with the OSS ecosystem's and how to build their own influence and position the ecosystem's governance structure [10]. The knowledge output can then be leveraged towards other stakeholders through the politics and negotiations that take place in the ecosystem's RE process [146].

These aspects highlight the importance of stakeholder identification and analysis as input to the continuous and complex decision-making process which RE constitutes [8] by helping to answer questions as which other stakeholders exist in the ecosystem, what are their agendas, and how do they aim to achieve them [65]. However, current practices [177] are not adapted to consider these strategic aspects [62] in the context of OSS ecosystem [159] and its informal and collaborative RE process [54, 192], specifically the importance of understanding stakeholders' influence and interactions. Involved firms are no longer the vantage point, and instead, form part of a larger set of interdependent stakeholders [189]. We address this gap with a design science research approach [90, 228] and define it as a design problem [228]:

DP *How to characterize the influence of stakeholders on the OSS ecosystem's RE process, so that a focal firm can understand other stakeholders' agendas, collaborations, and resource investments in pursuing these agendas?*

The contribution of our work is the proposal of the Stakeholder Influence Analysis (SIA) method. Its aim is to help firms to analyze an OSS ecosystem to identify its stakeholders' influence by the impact they have with respect to the requirements that get implemented in the OSS. We base SIA on social network analysis constructs [55, 164, 215] that have proven to be useful in characterizing the influence of stakeholders [176, 189], but also effective when analyzing a firm's participation in OSS ecosystems [176, 208] and requirement-centric stakeholder collaborations [17, 43, 142]. An analysis approach used in an earlier reported case study of the Apache Hadoop OSS ecosystem [132] is formalized to consider

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

how requirements may be informally represented in multiple artifacts in decentralized repositories present in OSS ecosystems [54, 192]. The influence analysis is then operationalized with a stakeholder mapping approach based on earlier work [104, 144, 162]. To demonstrate SIA's applicability and utility, we present a case study of the Apache Hadoop OSS ecosystem.

The rest of this paper is structured as follows: In section 2 we describe the research approach used in the development of SIA. In section 3 we give a detailed presentation of SIA, while in section 4 we demonstrate its applicability and utility with a case study. In section 5 we discuss alternative approaches to characterizing influence and threats to validity. Finally, we conclude the paper in section 6.

2 Research Approach

To develop SIA, we used a design science research approach [90, 228] where research is conducted iteratively through design cycles. A design cycle consists of three phases: problem investigation, artifact design, and artifact validation [228]. Below we describe these steps in detail.

Problem Investigation phase: Here, the research goal and the problem context are (re-)analyzed before any artifact is designed, or any improvements implemented [228]. In previous work [132], we explored how centrality measures could be used to characterize the influence of stakeholders within an OSS ecosystem, and how this evolved over time. Findings helped to create an understanding of the problem context and helped define the design problem (**DP**) as stated in Section 1. In order to further understand the problem context, a literature survey was conducted to identify related work on:

- the informal and collaborative RE processes within OSS ecosystems (e.g., [54, 101, 120, 161, 192]),
- how awareness of the dynamics behind stakeholder interactions and interrelationships may be used to analyze their agendas (e.g., [10, 65, 147, 157, 159, 177, 189]), and
- how social network constructs may be used to characterize the stakeholders' interactions and influence on the RE process of the OSS ecosystem (e.g., [12, 17, 42, 55, 142, 164, 176, 189, 208, 215]).

Surveyed literature provided conceptual foundations, which together with findings from previous work [132], constituted a knowledge base for the artifact design process.

Artifact Design phase: Here, knowledge gained from the previous phase is used as input to the design of an artifact with the hypothesis that it may act as a treatment for the design problem [228]. The Stakeholder Influence Analysis (SIA) method was formalized and structured as seven steps, as presented in Section 3

(S1-S7) and in Fig 1. S1-S2 involves setting the purpose and scope of the analysis. S3 concerns data gathering, while S4-S6 concerns data processing. Finally, S7 regards the analysis of the processed data.

Artifact Validation phase: Here, the previously designed artifact is tested in the problem context in order to evaluate its treatment of the design problem [228]. To test SIA, we apply it in a proof of concept demonstration that it is functional and practical, through a case study on the Apache Hadoop OSS ecosystem (see Section 4). It can be seen as an early form of descriptive validation where information from the knowledge base, and detailed scenarios can be used to demonstrate an artifact's applicability and utility [90]. The Apache Hadoop OSS ecosystem was chosen due to the high concentration of firms in the ecosystem, and because it is the Apache project with the highest number of committers [2]. The case study further helped to evolve and refine SIA and its seven steps as can be expected by an iterative design process.

3 The Stakeholder Influence Analysis (SIA) method

SIA aims to help firms involved in OSS ecosystems to structure their stakeholder identification and analysis process systematically when bridging their internal RE process with that of the ecosystem's (see Fig. 1). The focus is specifically on identifying and characterizing stakeholders' interactions and influence on the RE process in the OSS ecosystem. As proposed by Glinz and Wieringa [77], SIA considers both individuals and organizations as stakeholders but primarily from an organizational level, meaning that the individuals in an OSS ecosystem should be aggregated to their organizational affiliation as far as possible. Below, we give a detailed overview of SIA and its seven steps, as outlined in Fig. 1 and table 1.

Determine the purpose of the analysis process (S1): The first step 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 to identify conflicting agendas in regards to certain requirements.

Limit the analysis' scope based on its 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. E.g., is the analysis limited to:

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

[2] https://projects.apache.org/projects.html?number

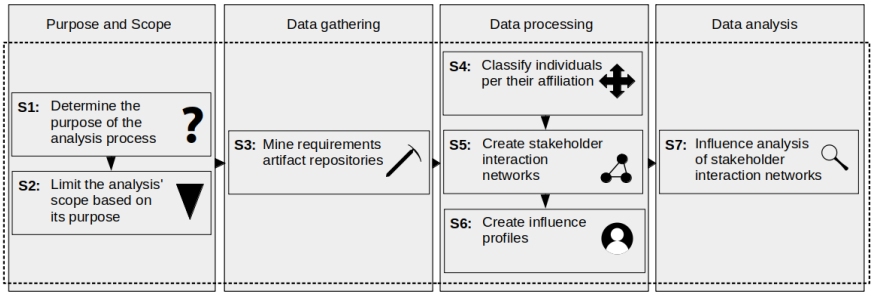


Figure 1: Overview of SIA's seven steps (S1-S7) divided in Purpose and Scope, Data gathering, processing and analysis.

Table 1: Overview of SIA and its seven sequential steps (S1-S7) along with related descriptions and examples.

	Step	Description
S1	Determine the purpose of the analysis process.	Purpose could include: • Understand how an ecosystem is set up in terms of power-structure and general collaboration patterns. • Identify potential partners or competitors as input to contribution strategies or collaborations. • Identify stakeholders with aligning or conflicting agendas in regard RE-related activities and negotiations. • Identify influential stakeholders to learn from in order to raise one's own influence in the OSS ecosystem.
S2	Limit the analysis' scope based on its 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 component or set of features of the OSS? • a certain individual or set of stakeholders? • a certain time-period or set of releases?

S3	Mine 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	Classify individuals per their 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. If no affiliation can be found or exists, the individuals can either be considered either as individual stakeholders or as an aggregated group.
S5	Create stakeholder interaction networks.	For each requirement artifact repository, a directed and weighted affiliation-network is created. Stakeholders are represented 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 based on the size of each stakeholder's participation.

S6	Create influence profiles.	To characterize stakeholders' influence on the RE process in the OSS ecosystem, a set of network centrality measures are calculated based on the interaction networks, and used to create an overall influence score. Together, they form an influence profile for each stakeholder. The centrality measures include: • Out-degree centrality • Betweenness centrality • Closeness centrality • Eigenvector centrality
S7	Influence analysis of stakeholder interaction networks.	Based on influence profiles, stakeholders are ranked on overall influence score, and cross-compared on the centrality measures. Stakeholders of special interest are investigated further in regards to their relationships. With qualitative analysis of stakeholders' agenda alignment with the focal firm's, stakeholder mapping can be used with the influence/alignment matrix. The analysis should be directed by the purpose defined in **S1**.

Mine requirements artifact repositories (S3): In the third step, the goal is *to identify and mine the repositories that are mainly used by the OSS ecosystem*. Examples include issue trackers, mailing-lists, IRC logs, source code repositories, and code-reviews [54, 192]. When these are identified, the repositories should be mined to collect the necessary data. This can either be done either manually or with the help of existing[3] or custom-made tools.

Classify individuals per their affiliation (S4): In the fourth step, the *individuals that are involved in OSS ecosystem need to be classified in regards to their affiliation*. This is a necessary step as firm-affiliated individuals may be assumed to represent the agenda of their sponsor or employer [40, 87]. 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) [19, 78, 132].

[3] See e.g., https://metricsgrimoire.github.io/

If no affiliation can be found or exists, the individuals can either be considered as individual stakeholders or as an aggregated group. *For example*, say, John, Mark, Lucy, Kate, and Mary are involved in the Apache Hadoop OSS ecosystem as developers. John and Kate work for a firm called Hortonworks and therefore have a common agenda. They are therefore aggregated and viewed as one stakeholder represented by the firm Hortonworks. Mark, Lucy, and Mary are all independent with the difference that Lucy is a relatively active user in the ecosystem, while Mark and Mary are more involved on a hobby basis. Lucy could, therefore, be seen as an independent stakeholder, while Mark and Mary could be aggregated to one group of hobbyists and be considered as one stakeholder. This type of classification and separation is rather subjective and needs to be done on a case-by-case basis for each ecosystem.

Create stakeholder interaction networks (S5): In the following step, *an interaction network for each requirements artifact repository needs to be created* in order to visualize the interactions between stakeholders. To create these networks, the interactions between the stakeholders to the requirement artifacts within a requirements artifact repository must be identified. As an example, consider a number of individuals (stakeholders) that discuss the need for as well as potential implementations of a new feature in an OSS project. The feature request is represented by an issue (requirements artifact) on the OSS ecosystem's issue tracker (requirements artifact repository). The discussions (interactions) between the individuals concerning the feature's evolution and refinement is recorded and persisted in the issue. This continuous discussion may be referred to as an "event" in social network theory [215]. The individuals partaking in the discussions may be referred to as "participants" of the same event [215].

These events and their participants can furthermore be represented by networks of actors. Two actors within a network are connected by an edge if they have participated in the same event (as a network may include several events). If a network was created based on the previous example, all individuals who partook in the discussion of the issue would be represented by an actor in a network with an edge connecting each one of them. If there was a related discussion of the feature on the OSS ecosystem's mailing-list, a similar network may be created based on the concerned mail-thread. The two networks could then be analyzed in conjunction to get a more complete overview of the stakeholders to the requirement and their interactions (cf. requirement-central networks [43]).

In a similar fashion, sets of requirements may be analyzed by aggregating requirements artifacts in a repository to a network. Returning to the example, a network could be created that included all of the issues in the issue tracker that are related to a certain release, created in a certain time span, or belonging to the same sub-module. A corresponding network could be created based on the mailing-list given that the same conditions apply. By creating corresponding networks of all the relevant requirements artifact repositories, the analyst may get a complete overview of what stakeholders that are involved and how they interact.

It should be noted that one stakeholder's participation in the event (e.g., RE-related discussions of an issue) 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 the 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. [176], weights can be calculated to describe the relative size of the participation to an event.

Following Orucevic-Alagic et al. [176], for a set of stakeholders $V = \{v_1, v_2, ..., v_k\}$ and a set of requirements artifacts (events) $U = \{u_1, u_2, ..., u_m\}$, we define a weight W of an edge between one stakeholder v_i and all other stakeholders that collaborate on an artifact u_t as:

$$W(v_i, u_t) = \frac{X(v_i, u_t)}{\sum_{c=1}^{k} X(v_c, u_t)}$$

where $X(v_i, u_t)$ denotes the number of times a stakeholder v_i has participated in the collaboration on the requirements artifact u_t.

Continuing from Orucevic-Alagic et al. [176], this means that the weight of the edge $W(v_i, v_j)$ for all requirements artifacts that two stakeholders v_i and v_j have collaborated on together equals:

$$W(v_i, v_j) = \sum_{t=1}^{m} W(v_i, v_j, u_t)$$

As an example, when creating an interaction network based on an issue-tracker, each issue represents a requirements artifact and number of posted comments may represent the size of participation (X) of a stakeholder. Given that three stakeholders v_A, v_B and v_C comment on the issue, they are all considered as actors in a network with edges connecting them. The weights would, therefore, consider the relative number of comments of each stakeholder as the size of their participation. Say v_A commented 1, v_B commented 2, and v_C commented 3 times. This results in the edge weights:

- $W(v_A, v_B) \& W(v_A, v_C) = 1/5$
- $W(v_B, v_A) \& W(v_B, v_C) = 2/5$
- $W(v_C, v_A) \& W(v_C, v_B) = 3/5$

If two stakeholder participated in an equal number of times, the 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 size of a stakeholder's participation (X) can be quantified with the

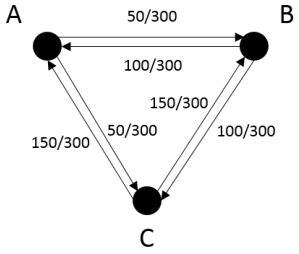


Figure 2: Example of network with three stakeholders v_A, v_B and v_C, and connecting weighted edges. Adopted from [132].

number of changed lines of code (LOC) of its patches. A simplified example is shown in Fig. 2 where three stakeholders v_A, v_B and v_C each created various number of patches that were contributed to a certain issue. v_A's patches contain 50 LOC in total. v_B's patches contain 100 LOC in total, while v_C's patches contain 150 LOC in total. Aggregated, 300 LOC were contributed to the issue. Resulting in the following edge weights:

- $W(v_A, v_B) \& W(v_A, v_C) = 50/300$
- $W(v_B, v_A) \& W(v_B, v_C) = 100/300$
- $W(v_C, v_A) \& W(v_C, v_B) = 150/300$

By constructing this kind of networks (i.e., weighted and directed affiliation-networks [55, 215]), 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**.

Create influence profiles (S6): In a network, a stakeholder is more prominent if it has a central position with edges that make it extra visible and important to others [12]. In social networks, centrality measures are commonly used to analyze an actor's position and prominence relative to others [215]. Faust [55] breaks down the notion of centrality into 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 flows 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 out-degree, betweenness, closeness and eigenvector centrality. SIA uses these measures as the foundation for analyzing the influence of stakeholders.

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 stakeholders have

weights attached to them. These weights allow the measures to take account of the relative size of each stakeholder's participation of the requirements artifacts on which the network is based on. E.g., out-degree centrality (see table 2) refers to the sum of weights attached to outgoing edges from the focal stakeholder and its adjacent stakeholders [13]. This gives an overall number in regards to the size of 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 [176]. However, this way of measuring out-degree centrality does not provide information about the total number of connections of a stakeholder, which may better show the number of collaborations and opportunities to spread one's opinions [175]. 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 [61].

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

As described by Faust [55], centrality may be broken down into multiple aspects. Centrality measures, in turn, use different definitions and sets of criteria in regards to what classifies an actor's position as central. Hence, one measure can present a different social structure than another and different measures provide different perspectives on who are the most active [176]. In smaller and simpler network structures such measures may co-vary, while in larger and more complex networks, they may characterize actors very differently [81].

Therefore, measures presented in table 2 could be seen as complementary to each other and may be used together to give each stakeholder (v_i) an *influence profile* (IP_{v_i}), a 4-tuple consisting of each centrality measure (i.e., out-degree centrality (ODC_{v_i}), betweenness centrality (BC_{v_i}), closeness centrality (CC_{v_i}), eigenvector centrality (EC_{v_i})).

$$IP_{v_i} = (ODC_{v_i}, BC_{v_i}, CC_{v_i}, EC_{v_i})$$

Such a profile can then be used when analyzing a stakeholder's interaction network in step **S7**. E.g., a stakeholder in a certain interaction network may have

- a high ODC indicating a high activity with many collaborations,
- a low BC indicating that the stakeholder does not have a broker's position, but
- a high CC indicating that the stakeholder can more easily reach out with its communication, and
- a high EC indicating that the stakeholder knows other influential stakeholders.

When comparing stakeholders and their influence profiles, it would be convenient to define, for each stakeholder v_i, an aggregated *influence score* IS_{v_i}. Such a score could be used to divide stakeholders in to two groups, those with a high and low level of influence (see upper and lower zones in Fig. 3). One way to do this aggregation is to simply add the normalized weights of each element in the profile, resulting in a ratio-scale number between 0 and 1, as given by the formula below, and then group stakeholders based on a threshold, eg. less than or equal to 0.5 denotes low influence:

$$IS_{v_i} = \frac{1}{4}\left(\frac{ODC_{v_i}}{ODC_{max}} + \frac{BC_{v_i}}{BC_{max}} + \frac{CC_{v_i}}{CC_{max}} + \frac{EC_{v_i}}{EC_{max}}\right)$$

There are other ways of aggregating the different measures, using e.g. ordinal-scale ranks, a vector space distance metric (e.g. cosine similarity), a normalized exponential function (softmax), or applying some kind of weighting scheme to reflect e.g. that centrality is considered more interesting. Another option is to qualitatively compare the IS_{v_i} 4-tuple of measures in combination with some visualization technique, such as spider diagrams or similar. Future work should investigate which aggregation method that would best help to partition the stakeholders into high- and low-level category.

In addition to comparing the stakeholders' influence profiles and overall influence scores within a specific stakeholder interaction network, it is equally important to compare between the networks. For example, if the analysis includes multiple requirements artifact repositories (e.g., issue-trackers and mailing-lists) or covers multiple releases, these could be cross-compared. A stakeholder may have a high overall influence score in one requirements artifact repository, and less in another. Further, the influence and interactions may shift with time why temporal analysis may give important insights. Also, it may be that one repository is more important than another (e.g., issue-tracker over mailing-list), as a result, the former should be given more attention in a cross-comparative analysis of a stakeholder.

Influence analysis of stakeholder interaction networks (S7): In the influence analysis, the interaction networks and influence profiles from **S5** and **S6** are used to address the purpose defined in **S1**. First, stakeholders are ranked on their overall influence score to get an overview of the stakeholder population. Stakeholders of interest, e.g., a top-list of those most influential, can then be cross-compared based on the centrality measures from their influence profiles, and analyzed in detail, e.g., in regards to their relationships. Table 2 provides descriptions of how the centrality measures may be interpreted in terms of a stakeholder's influence in the RE process of an OSS ecosystem.

As a support in the analysis, and to help address the purpose as defined in **S1**, stakeholder mapping can be applied with the use of an influence/agenda alignment matrix (see Fig. 3). The matrix, based on earlier work [104, 144, 162], is adapted to consider the power and politics [65] that play a central part in the RE process of

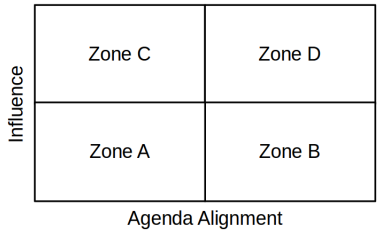


Figure 3: Influence/Agenda alignment matrix to be used for stakeholder mapping. Adapted from earlier work [104].

OSS ecosystems [157, 159]. The Y-axis represents the level of influence and the X-axis how well their agenda in the OSS ecosystem aligns with that of the focal firm. Both dimensions range from low to high. The four quadrants Zone A-D in the figure are explained subsequently.

The level of influence of a stakeholder is based on the influence score from **S6**. The threshold for when a stakeholder's influence score ranks as high is set by the analyst in relation to the total number of stakeholders in the network. Agenda alignment, which is the second dimension, is determined by qualitatively investigating the previously identified stakeholders' engagement in the OSS ecosystem, e.g., by reviewing comments made by the stakeholder in the set of issues which the analysis considers (as defined in **S1** and **S2**). The investigation should answer if the stakeholder and the focal firm want the same thing, and to what extent.

The classification puts a stakeholder into one of four quadrants (A-D) of Fig. 3, each indicating a different relationship and possible engagement that the focal firm should establish and maintain with the stakeholder. Stakeholders with a high level of influence and high level of agenda alignment (Zone D) may pose as (potential) partners, both in regards to general collaboration and RE related activities and negotiations. Stakeholders with a high level of influence and low level of agenda alignment (Zone C) may pose as the key opponents and may require active engagement in negotiations in the RE process of the OSS ecosystem. Stakeholders with a low level of influence (Zone B and A) may not pose as having high importance, but may still require monitoring as they can move their position with time. Those in Zone B may pose as future collaboration opportunities, while those in Zone A as potential threats.

If competitors are identified among those with high influence, this may signal that they have a high interest in the ecosystem and scope of the investigation. If they are found in Zone D, there might be an opportunity for co-opetition. In either case, whether they have aligning agendas or not, consideration should still be taken to the differential value of what is contributed and how resources are invested. By studying stakeholders in Zone C and D, a focal firm can potentially strengthen its own influence by learning from these stakeholders, in how they invest their resources and with whom they collaborate. This may lead to further collaboration and other potential partners, and how interest may overlap between multiple stakeholders.

Table 2: Network measures described from a general perspective as well as and how they can be interpreted from a RE perspective in regards to Stakeholder influence.

Measure	Description	Stakeholder Influence Interpretation
Out-degree Centrality	Refers to how well-connected the focal actor is and considers the outgoing edges towards its adjacent actors, where the focal actor is the transmitter (source) for the edges. With weights considered, this measure refers to the sum of weights attached to the outgoing edges of the focal actor [13]. With binary edges considered, this measure refers to the number of outgoing edges between the focal actor and its adjacent actors [61].	Out-degree centrality is generally considered as a measure of activity that can identify "where the action is" and highlight the most visible actor in the network [215]. With weights considered, a high out-degree centrality is an indication of influence on adjacent stakeholders as the focal stakeholder has participated in a large part in the requirement artifacts which they have interacted with [176]. 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 a number of collaborations, but also to which the focal stakeholder has expressed its opinions.

Betweenness 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 the lowest sum of weights [23, 163]. With binary edges considered, it refers to the shortest path in regards to the least number of edges [61].	Betweenness 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 [215]. Hence, stakeholders with a high betweenness centrality 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 whom it reaches and when, to better serve personal priorities. When a stakeholder 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 [142, 164].

Closeness Centrality	Refers to the inverse of the sum of the shortest paths from the focal actor to all others in the network. With weighted edges considered, it refers to the shortest path with the lowest sum of weights [23, 163]. With binary edges considered, it refers to the shortest path in regards to the least number of edges [61]. This measure only considers those actors that are connected to the same network as the focal actor [164]. 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 [164]. Hence, 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 [189].
Eigenvector Centrality	Refers to how connected an actor is, similar to out-degree centrality, but considers how well-connected the adjacent actors are [21]. The focal actor receives a score based on a sum of its adjacent actors' scores [164].	Eigenvector centrality is a measure of activity and visibility as out-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 [164]. Hence, 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 [55]. 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 actors in the social network [55].

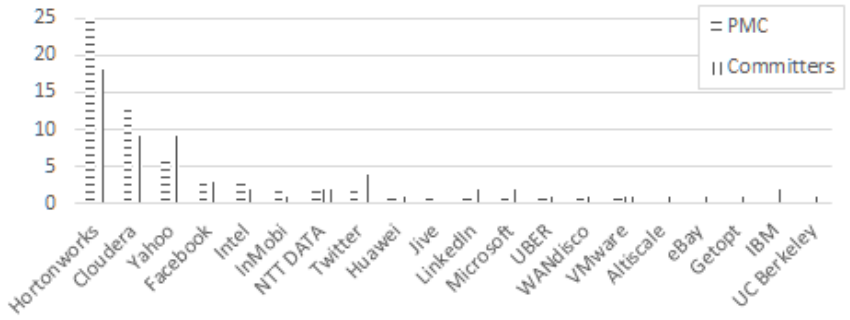


Figure 4: Number of committers and members in the Apache Hadoop PMC aggregated per firm.

4 Case Study of Apache Hadoop OSS Ecosystem

In this section, we describe a first evaluation of SIA in our design methodology. We demonstrate the applicability and utility of SIA in a case study [191] on the Apache Hadoop OSS ecosystem. The case study takes the perspective of a (fictive) focal firm that provides scalable and secure infrastructure on which Hadoop can be deployed for customers. This is a new product offering, and the focal firm is now interested in becoming active in the Apache Hadoop OSS ecosystem. As they are new to the ecosystem, they want to do an initial stakeholder analysis to see if there are any potential partners to collaborate with, and potentially learn from (**S1**). First, they want to get a general overview of the stakeholder population to see who is present and how the ecosystem functions in terms of the power structure and collaboration patterns. Second, they will look for potential partners among those most influential and investigate how they work, and what interests they have in the ecosystem.

The Apache Hadoop project[4] is a widely adopted OSS framework for distribution and process parallelization of large data, originating from *Yahoo* in 2006. The framework consists of four modules: Hadoop Common Modules, Hadoop Distributed File System (HDFS), Hadoop YARN, and Hadoop MapReduce.

The Apache Hadoop project is part of the Apache Software Foundation which is an umbrella organization for a large number of OSS projects and their ecosystems. A common trait for these projects is the use of meritocracy in terms of culture and governance[5]. This is reflected in the governance structure among the Apache projects, as in Apache Hadoop which is governed by a Program Manage-

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

[5] https://www.apache.org/foundation/how-it-works.html#meritocracy

ment 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 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 in the development of the project. Hence, power may be earned by showing a long-term commitment and having the competence needed (i.e., meritocracy). In Fig. 4 the distribution of members of the committers and the PMC are presented based on affiliation per firm.

4.1 Overview of Stakeholder Interaction and Influence

To get a recent view on who the most influential stakeholders are, the scope of the analysis is limited to requirements included from release 2.2.0 (15/Oct/13) to 2.7.1 (06/Jul/15) (**S2**). To get a view on both social and technical interaction, the issue-tracker is analyzed in regards to requirements artifact repositories (**S3**). The issues contain both comments (the social dimension) and patches (technical). The patches are committed by authorized users, once they have been approved. To identify the organizational affiliation of individuals that have interacted via the requirements (**S4**), an analysis is done of e-mail subdomains, complemented with cross-checking against other information sources (e.g., social media and electronic archives) [19, 78, 132]. For a subset of individuals, an organizational affiliation could not be determined. These individuals were aggregated into two separate groups, either independent (if this could be determined) or unidentified.

Creation of Stakeholder Interaction Networks (S5): Based on the scope specified in **S2**, and the repository identified in **S3**, two interaction networks are generated: a *comments-network* to include stakeholders who commented on common issues, and a *patch-network* to include the stakeholders who contributed patches to the same issues (**S5**). The patch-network was presented in earlier work [132], and a similar data collection and cleaning approach were used in order to create the comments-network, as is also proposed in SIA (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 3). 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 collaborations per stake-

holder, compared to the patch-network, which has an average of 3 collaborations per stakeholder. Both networks are visualized on a high level in Fig. 5 and 6. Labels are of firms and of relative size to their weighted out-degree, a reason for which only those with the highest values may be readable.

Table 3: Characteristics of comments- and patch-networks.

	Comments-network	Patch-network
Stakeholders	122	86
Collaborations	1096	260
Per stakeholder	9	3

Creation of influence profiles. (S6): To measure the influence of, and collaboration among, the stakeholders (**S6**), two SNA measures were leveraged: weighted out-degree and betweenness centrality. Other centrality measures presented in table 2 were excluded due to space considerations in this paper. In Fig. 7 and 8, the two measures are presented in two separate diagrams. The diagrams contrast the respective measures for the comments- and patch-networks in regards to the 15 top stakeholders (considering the overall influence score).

Influence Analysis of the Stakeholder Interaction Networks (S7): As presented in table 2, the measures measure different aspects of influence and collaboration among the stakeholders. Below, the two measures are compared in regards to the two networks and their stakeholders.

Out-degree centrality: Fig. 7 illustrates the normalized out-degree centrality which may be considered as 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.

Betweeness centrality: In Fig. 8, it can be seen that the normalized betweenness centrality varies notably between the comments- and patch-networks for the top stakeholders. Hortonworks has the highest betweenness centrality in regards to both the technical and social aspects and compared to Cloudera and Yahoo, it has double the betweenness 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.

Cross-comparison of centrality measures: To simplify the cross-comparison, the influence score is used to get an overview of the top 10 most influential stake-

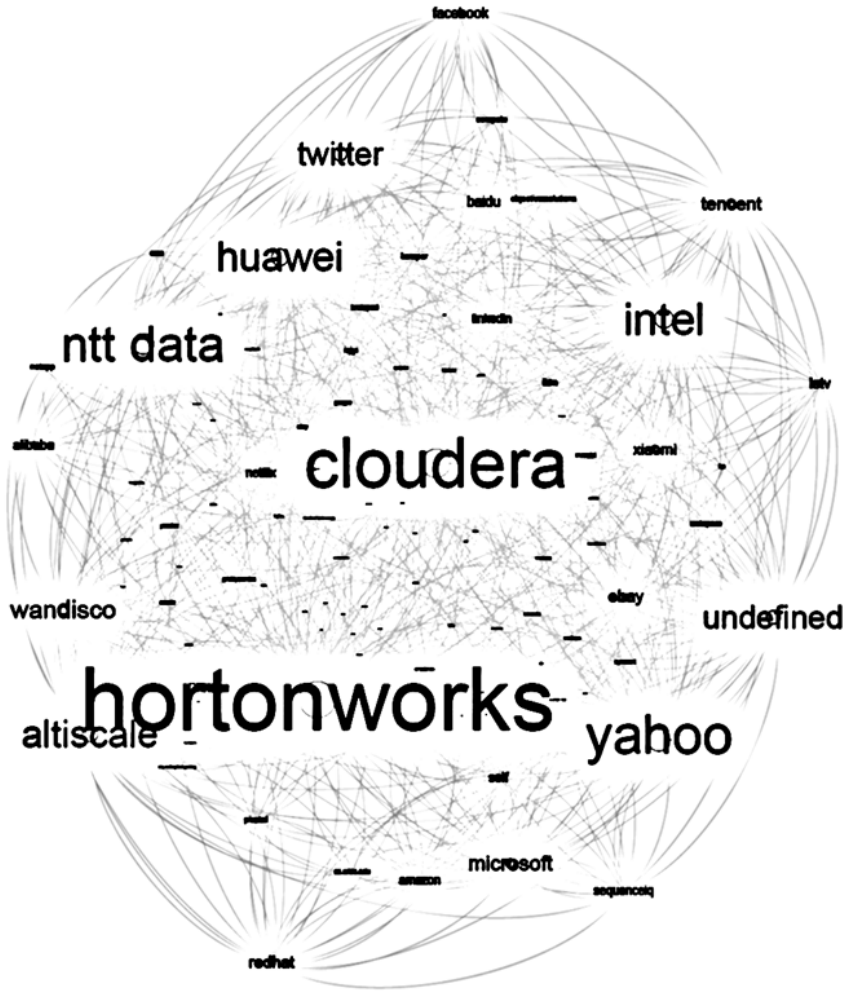


Figure 5: Visualization of the comments-network. Labels are of firms and of relative size of their weighted out-degree to other firms in each network.

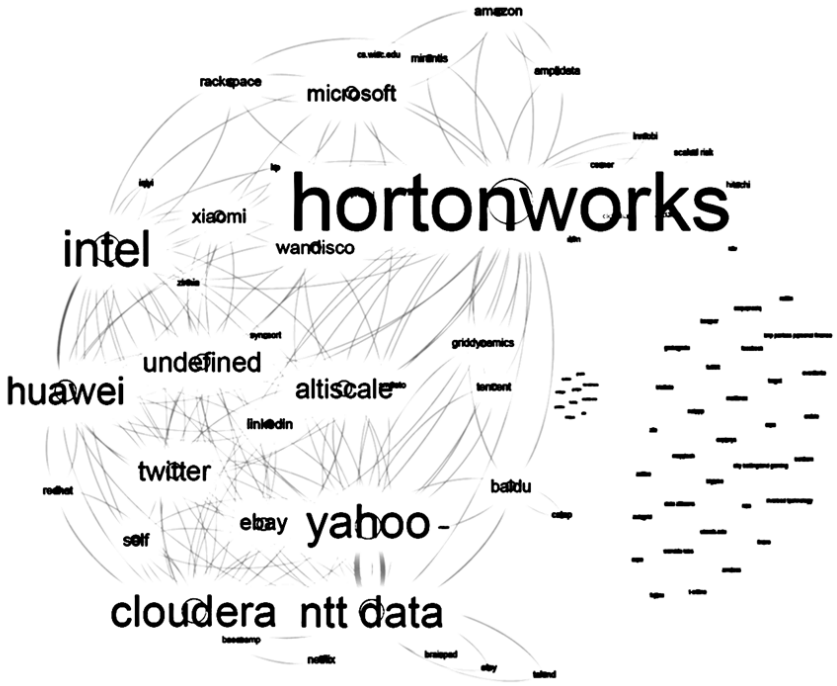


Figure 6: Visualization of the patch-network. Labels are of firms and of relative size of their weighted out-degree to other firms in each network.

Table 4: Top ten stakeholders based on influence score based on comment-network, considering only the out-degree and betweenness centrality.

Stakeholder	Outdegree (norm)	Betweeness (norm)	Influence score
hortonworks	0.66	0.86	0.76
cloudera	0.44	0.4	0.42
ntt data	0.28	0.22	0.25
huawei	0.26	.10	0.18
yahoo	0.25	0.10	0.18
intel	0.19	0.11	0.15
undefined	0.11	0.09	0.10
twitter	0.14	0.06	0.10
altiscale	0.11	0.07	0.09
wandisco	0.13	0.02	0.8

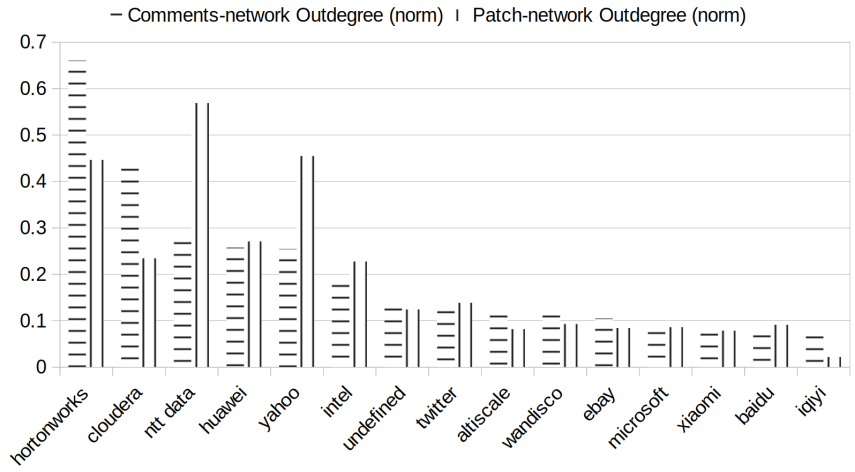


Figure 7: Visualizations of normalized out-degree centrality for the 15 top influential firms across the comments- and patch-networks. The diagram is sorted in a descending order based on out-degree centrality from the comments-network.

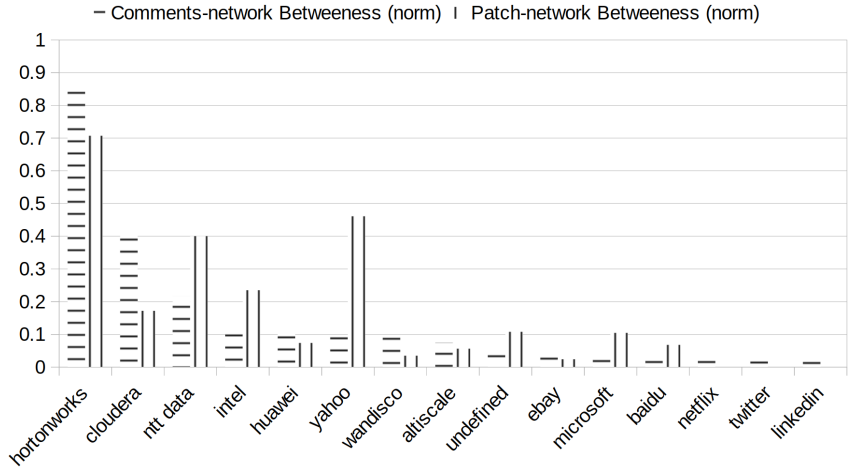


Figure 8: Visualizations of normalized betweeness centrality for the 15 top influential firms across the comments- and patch-networks. The diagram is sorted in a descending order based on betweeness centrality from the comments-network.

holders considering the two centrality measures (see table 4). Comparing the two centrality measures for these ten, both similarities and differences may be found. Although it has higher activity in the comments-network, Hortonworks has high influence in regards to both technical and social interaction, if both 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 both of the two 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.

Considering the influence/agenda alignment matrix (see Fig. 3), these stakeholders could be considered as key stakeholders and qualify for either Zone C or D. They could pose either as potential partners or threats depending on how their agenda aligns. Also, depending on if they are competitors or not, consideration should also be taken when constructing contribution strategies [231]. The focal firm should, therefore, monitor and form an understanding of how these stakeholders' agendas align with their own.

4.2 Investigating Collaborations and Agenda of a Potential Partner

From the previous analysis, the focal firm could identify WANdisco as a stakeholder with a similar business model and a potential partner in terms of collaboration and similar interests (Zone D in Fig. 3). The goal in this second step is to do a more thorough analysis focusing on WANdisco's collaborations and high-level agenda (**S7**).

Looking at WANdisco's influence profile, their overall influence score gives them an ordinal rank of 10 (see table 4) when analyzing the comments network. They have an equal level of social and technical activity, on similar levels as Twitter, Altiscale, eBay and Microsoft (see Fig. 7). They have a relatively high level of control and coordination considering the betweenness centrality. All things considered, they have a relatively high influence and interest in Apache Hadoop, but much lower than the key-stone players, Hortonworks, Cloudera, NTT Data, Huawei, Yahoo, and Intel.

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, similar to that of the focal firm. WANdisco has 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 learn more about WANdisco's interests in Apache Hadoop and its collaborators, the focal firm investigates if WANdisco has shown a special focus in regards to any of the four modules of Apache Hadoop: Common, HDFS, YARN, and MapReduce (**S2**). The analysis is still focused on requirements included in releases R2.2-R2.7. In regards **S3-4**, they are identical to the previous example. When creating the interaction networks (**S5**), one patch- and one comments-network are created for each of the modules.

When creating the influence profiles (**S6**), the analysis is limited to examining the out-degree to get a view of their activity and comprehension of their relative influence in regards to the modules. Values for binary and weighted out-degree are presented in table 5 and table 6 respectively. 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, it can be concluded that WANdisco has a specific interest in the HDFS module. The out-degree values for the comments network further confirm 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 of Wandisco for Apache Hadoops four modules. Values aggregated for releases R2.2-2.7 per network type. Relative ranking 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 of Wandisco for Apache Hadoops four modules. Values aggregated for releases R2.2-2.7 per network type. Relative ranking 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, the analysis is limited to the HDFS module as this

is where their main interest of WANdisco 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, Hortonworks, Cloudera, Intel, Pivotal and Yahoo were the top five in regards to the number of comments made by WANdisco on common issues, see table 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 have collaborated on.

An outcome of this analysis is that WANdisco holds their main interest and invest their resources in the HDFS component, both from a technical and social perspective. Considering the influence/agenda alignment matrix in **S7** (see Fig. 3), a qualitative investigation needs to be performed, e.g., of their comments and code commits, in order to determine e.g., what features they value or prioritize. Such an investigation will help to determine the agenda alignment further and if WANdisco belongs to Zone C or D, i.e., if they make up a potential opponent or partner. Based on their active collaborations, Pivotal should be investigated further in terms of their interest and activity.

Table 7: Top collaborators with Wandisco in the comments network of 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

5 Discussion

Below we first discuss different alternatives to characterizing a stakeholder's influence, followed by a discussion regarding limitations and threats to validity in our demonstration of SIA's utility in the analysis of the Apache Hadoop ecosystem stakeholder influence.

5.1 Alternatives to Characterizing a Stakeholder's Influence

The three questions stipulated by Frooman [65] highlights the strategic importance of stakeholder identification and analysis: 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 informs of a firm's possible strategies for contribution and interaction. SIA helps to address these questions by characterizing the stakeholders' collaboration and influence within the OSS ecosystem. The quantitative outcome which is generated is best complemented with qualitative insights which may be gained through observing or even taking part in the communication of the ecosystem.

In the stakeholder mapping process, which is part of the influence analysis of SIA (**S7**), both the quantitative and qualitative aspects are needed. In the proposed influence/agenda alignment matrix, the influence profiles and influence scores may be used to determine the level of influence. As mentioned i Section 3 and the description of (**S6**), the proposed influence score is one approach to get a simplified overview of which stakeholders are the most influential. However, as the different centrality measures provide different aspects, these measures should still be investigated qualitatively to get a more fair view over how influential the stakeholders can be considered to be relative others.

On a general level, one can also look at which stakeholders hold a seat in the different committees of the ecosystem governance structure. However, these do not have to align as a firm can influence both by having representatives in and outside leadership positions, of which the latter is the more common [194]. This phenomenon can be noted when comparing firms with members on the PMC and Committers group in the Apache Hadoop OSS ecosystem (see Fig. 4) with firms that have the overall highest influence score (based on outdegree and betweenness centrality, see table 4). NTT Data, with a relatively high activity both in regards to the patch and comments networks (see Fig. 5 and 6), have a very limited number of places on both the PMC and Committers group. Furthermore, when changing the scope of the analysis (e.g., a certain set of releases or a component - see Section 4.2), the governance structure may give an even less representative overview as different stakeholders have different interests, why the network approach proposed by SIA may prove more valuable.

Another approach to measuring influence with other than centrality measures would be to use pure count-based measures of a developer's activity, e.g., number of comments and code-commits. As highlighted by Joblin et al. [103], these, however, give a simplified view of a developers position and do not consider the inter-developer relationship. By considering the latter, an analysis can investigate, for example, how active the developers are, how efficiently they can communicate with others, how they are able to mediate and control the flow of information be-

tween others, and have relationships "with others that are themselves" central [55]. As further shown [103], network-based measures are equally good, and in certain cases better than count-based measures at describing how influential a developer is. Certain count-based aspects are however included in SIA as it does recommend the use of binary edges as a complement to the weighted edges. As is mentioned in S6 (see Section 3), a high out-degree centrality based on weighted edges 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 [176]. However, this way of measuring out-degree centrality does not provide information about the total number of connections of a stakeholder, which may better show the number of collaborations and opportunities to spread one's opinions [175].

Furthermore, it may be noted that there are other centrality measures available [215] than those proposed in the CSF. We based our choice of out-degree, betweenness, closeness and eigenvector centrality measures on the suggestion of Faust [55] as explained in section 3. These are generally adopted in explaining the centrality and importance of an actor when analyzing OSS ecosystems (e.g., [18, 92, 176]).

5.2 Limitations and Threats to Validity

As a proof of concept demonstration that SIA is functional and practical in stakeholder analysis in a large ecosystem, we described a case study on the Apache Hadoop OSS ecosystem. The ecosystem has a community-managed governance model, meaning that the OSS project is owned and managed by the community [172], and a meritocratic authority structure, meaning that influence is gained by proving merit [161] and by establishing a symbiotic relationship with the ecosystem [38]. Another important characteristic of the chosen ecosystem is the high concentration of firms among its stakeholders, as we are interested in identifying and analyzing stakeholder on the organizational level.

This application of SIA in our case study, however, is not without a number of threats to validity. A threat to the internal validity concerns the way how weights are calculated (see S5, Section 3). The consideration taken to the relative size in regards to changed lines of code does account for the net amount (i.e., added and removed lines), but commits containing larger amounts of non-meaningful content may give a non-fair view. Thus it may be valuable to compare interaction networks and influence profiles based on both weighted and binary edges. Also, comparing networks based on different requirement artifact repositories (as exemplified in Section 4) can help to give a more nuanced view.

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 [54]. Requirements consist of fragmented representations, such as issues, mail-thread discussions and commits [192]. Further mitigation of 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 the scope of SIA as we focus on the identification and 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 process in future research.

A further threat concerns the determination of organizational affiliation of individuals in the OSS ecosystem. We adopted a heuristic approach as suggested by earlier research [19, 78], 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 SIA, we recommend using a mix-method triangulation with both qualitative and quantitative approaches.

The case study we described exemplifies how the different steps of SIA can be applied and the insights that can be gained. We acknowledge that a single case study is not sufficient to prove validity in terms of repeatability and utility, and that this is only a first step in the artifact validation phase of our design science research [228]. The characteristics of the Apache Hadoop OSS ecosystem, i.e. community-managed, meritocratic and multi-vendor, do however indicate what types of OSS ecosystems might benefit from a stakeholder analysis using SIA. Further investigation of SIA's utility and repeatability is out of the scope of this study and instead left for future work. Future research should consider applying SIA from a focal firm's perspective and study different types of OSS ecosystems with a more nuanced authority structure, e.g., as both autocratic, democratic and meritocratic coordination processes can act in parallel [198]. This falls naturally in the design science research approach as it is an iterative search process for an artifact that will solve the stated problem [90].

6 Conclusions

This study proposes the Stakeholder Influence Analysis (SIA) method which aims to help firms involved in OSS ecosystems to characterize ecosystem's 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, and often meritocratic governance structure. SIA, therefore, allows firms to see in which requirements a stakeholder holds a certain interest, and thereby create an overview of a stakeholder's agenda. This also allows firms to understand how stakeholders invest their resources, and with whom they collaborate according to their agenda. Thus, SIA offers input to how firms involved in

OSS ecosystems should construct their contribution strategies and act in the politics and negotiations of the ecosystem's RE process in order to align it with their internal RE process and product planning. It can be concluded that SIA shows potential through the case study on the Apache Hadoop OSS ecosystem, while further work is needed in regards to external validity.

In future work, we therefore aim to refine and validate SIA quantitatively and qualitatively through further design cycles involving additional OSS ecosystems, and from existing focal firms' perspectives. Further, we aim to investigate how the stakeholder analysis processes resulting from SIA may be used as an input to the construction and execution of contribution strategies [231].

Acknowledgements: We would like to thank Dr. Alma Orucevic-Alagic and the anonymous reviewers for their valuable feedback and inputs.

CHAPTER V

A CONTRIBUTION MANAGEMENT FRAMEWORK – WHAT TO SHARE AS OPEN SOURCE SOFTWARE?

Johan Linåker and Björn Regnell.

Abstract

Context: Software-intensive organizations' rationale for sharing Open Source Software (OSS) may be driven by both idealistic, strategic and commercial reasons, and include both monetary as well as non-monetary benefits. To gain the potential benefits, an organization may need to consider what they share and how, as a number of risks, costs and other complexities may interplay. **Objective:** This study aims to help organizations identify relevant objectives and complexities and thereby develop contribution strategies that describe *what* to contribute, *where* and *when*, aligned with the organizations' business goals. **Method:** A design science research approach is used, continuing from earlier research by performing a second and third design cycle with a multiple-case study of three case organizations. Gathered data includes interviews with 20 practitioners from the case organizations and contribution request forms from seven organizations. **Results:** To achieve alignment between business goals and contribution strategies, this study proposes a contribution management framework for software-intensive organizations to use in order to develop contribution strategies as well as contribution strategy guidelines and related contribution processes for internally developed software artifacts. The framework consists of contribution objectives and complexities that an organization may consider and weigh against each other when deciding on a contribution strategy. The framework further presents template questions, which can be used to tailor a contribution request form to an organization's specific objectives and complexities. **Conclusions:** Cross-case analysis and interview validation show that

the framework may provide a useful tool for practitioners in setting up internal contribution processes.

1 Introduction

Sharing, or contributing, software artifacts (e.g., features, projects, and frameworks) as Open Source Software (OSS) is a common practice among software-intensive organizations today[1]. By "opening up" [34], an organization can exploit the external workforce residing in the OSS communities that develops and maintains the OSS. Improved product innovation, accelerated development, lower maintenance cost, and improved branding and reputation are some of the potential intangible benefits that may motivate [86,134,157,159,205]. The motive may also be driven by pure idealism and being a good OSS citizen [98]. For some organizations, OSS may have a more direct connection to the business model or strategy, e.g., as a basis for complementary products and services [6,197,200], or as a mean to create a new standard or compete with existing ones [86, 98, 134, 219]. Objectives for why an organization would choose to contribute software artifacts as OSS does, however, not have to be limited to one or the other [6]. For consistency, we introduce the term **contribution objective**, which we define as *a purpose for contributing a software artifact, motivated by a monetary or non-monetary benefit that is enabled or resulted directly or indirectly as a consequence the contribution*.

To gain the potential benefits defined in the contribution objectives, an organization needs to access the external workforce, either by contributing its software artifacts to an existing OSS community or by creating a new community, each with its respective costs and risks [39]. In either case, the organization then needs to work actively to align its internal strategy with the community where they are a stakeholder among many, potentially including competitors and ones with conflicting agendas [38, 40]. An organization, therefore, may have to consider not just where, but also what it contributes, and when. Risks include giving away differentiating functionality [86, 87, 96, 212, 222, 231], or contributing too late and having to choose between adopting the alternative solution or maintaining the internal one alone [128, 231]. Hence, there are several potential costs and risks tied to a contribution. For consistency, we refer to these as **contribution complexities** and define them as *aspects of or related to a software artifact that may complicate the contribution of the artifact, or imply a cost or risk as a result thereof, either directly or indirectly*. As with contribution objectives, not all complexities may be relevant for all organizations [128].

By not considering relevant contribution complexities or objectives, an organization may risk making a contribution that could be hurtful, or on the opposite, block a contribution that could have been beneficial for the organization and its

[1] https://slideshare.net/blackducksoftware/you-cant-live-without-open-source-results-from-the-open-source-360-survey

business goals [231]. To gain the expected benefits, organizations hence need to link their business goals with their decisions on what they contribute as OSS, adopting what Aurum and Wohlin [9] refers to as a value-based approach. Despite the problematic context, research on how organizations can develop and use such strategies is limited [159], with some exceptions [128, 217, 231]. This leads us to define the following research questions:

RQ1 What contribution *objectives* should a software-intensive organization consider when assessing if, where and when a software artifact should be shared as OSS?

RQ2 What contribution *complexities* should a software-intensive organization consider when assessing if, where and when a software artifact should be shared as OSS?

The paper addresses these research questions by presenting a framework for Contribution Management (subsequently denoted CoMn, pronounced *'common'*). To achieve alignment between business goals and contribution strategies, software-intensive organizations may use the CoMn framework to develop contribution strategy guidelines and related contribution processes for internally developed software artifacts. A software artifact as a unit can range in size and complexity from minor bugfixes and features to frameworks, projects, and products, including test cases and design documentation. A contribution strategy describes *what* should be contributed, *where*, and *when*. For a specific artifact, the "*what*" indicates if the artifact should be contributed in full or kept closed, or if certain parts can be contributed under certain conditions. The "*where*" indicates whether the artifact should be contributed to one of many existing OSS communities or if a new community should be established. Finally, the "*when*" indicates when an artifact should be contributed in time.

More general contribution strategies can also be developed for generic types of software artifacts, e.g., bug-fixes and features to OSS communities, which are considered as not having a competitive advantage [157]. Contribution strategy guidelines describe the possible contribution objectives for why an organization would wish to share software as OSS, and what contribution complexities that may be relevant to consider when deciding on a contribution strategy for a specific artifact. The guidelines can further be used to set up a contribution process with tailored contribution request forms that can help decision-makers ask the right questions, and for the person filing the request, to better motivate why the request should be accepted.

The CoMn framework is created using a design science research approach [228] with three design cycles. Through collaboration with Sony Mobile, the first design cycle presented by Linåker et al. in earlier work [128] designed a Contribution Acceptance Process (CAP) model for creating contribution strategies [128]. The second and third design cycles cover a multiple-case study involving three case organizations (CaseOrg1-3) and is presented in this study.

The rest of this paper is structured as follows. Sections 2 presents related work to this study as well as previous work from which this study continues. Section 3 presents the research design while Section 4 provides background information to the three case organizations. Section 5 presents the CoMn framework, and section 6 provides a discussion on the framework in regards to related work as well as our our previous work. Section 7 presents a discussion on threats to validity, while section 8 gives the conclusions of the paper.

2 Related and Previous Work

This section provides an overview of related work to the two concepts of contribution objectives and complexities. This is followed by a presentation of our previous work with Sony Mobile [128, 151, 157] including an overview of the CAP model, which was designed in a previous design cycle [128], and how a related contribution process within an organization may be set-up. Finally, we present a summary of the related and previous work.

2.1 Benefits of Sharing Software as OSS

Several studies have systematically surveyed the literature and to different extents covered the benefits for why software should be shared as OSS [84, 93, 159, 197]. We categorize the benefits into four different themes.

A common theme is the cost-saving aspects [6, 158, 159]. By extending the resource-base [39] and agreeing on a common standard [222], organizations can share the maintenance and quality assurance, accelerate the development and potentially decrease their time-to-release and market [86, 91, 134, 137, 157, 205]. By freeing up internal resources, they can focus on more value-adding activities [134, 157, 212]. On the opposite, by adopting a less symbiotic relationship to the OSS community [38], an organization will have to maintain an internal branch of the OSS project, which may become costly depending on the number of modifications that need to be applied to new releases of the OSS project [213, 231]. Hence, to attain these potential benefits, an active engagement and symbiotic relationship may be needed with the OSS community [29, 40].

Another common theme is the innovation aspects [6, 159], which can be both product and process-oriented [158]. By opening up the innovation process [32] and ”pooling” the R&D/product development [222], organizations get access to an external workforce [136], which may bring increased knowledge sharing [137, 160] and innovation at a lower cost [205, 234]. However, this external workforce should be seen as a complement rather than a substitute for internal knowledge and development [39, 202]. Munir et al. [159] describe it as a catalyst for ideas that may help organizations in broadening their offerings. Hence, an organization may question how much of its internal R&D and innovation process it should outsource to a community [2].

A third theme can be tied to improved reputation [159, 213]. By creating a community or contributing to an existing community, an organization can create a marketing channel both towards (potential) customers, as well as future employees and have a positive effect on internal developers' satisfaction [39,86,136,185,205]. The improved reputation can turn into a competitive advantage [88] and legitimize the use of the OSS from a public perspective [40]. An organization's customers are offered an opportunity to avoid vendor-lockin, and the ability to customize the software to internal needs [137, 157].

A fourth theme concerns control aspects [128]. If an OSS community has a meritocratic coordination process in place [198], influence on the development direction of the community may be gained by participating in the development and maintaining a symbiotic relationship [24,38,40,129,166,194,206]. This may help steer the community including competitors and to manage potentially conflicting agendas [129, 138, 159, 194, 225].

Influence may also come implicitly when an organization's project or a feature is released and accepted as a standard solution, either within an existing community or as a new community [157]. If contributed to an existing community, other organizations will either have to accept and adapt, maintain internal forks of their own solutions, or attempt to contribute their solutions in competition with the solution already established within a community [128]. If released as a new community and traction is gained, it can potentially become a new standard or compete with existing [86, 98, 134, 219], and create a surrounding ecosystem with complements from other organizations [220].

Some of these themes may be more or less important depending on the type of organization. Munir et al. [158] present a theory of openness that categorizes organizations using OSS in their tools and infrastructure setups based on why and when they adopt and share software as OSS. The "why" is either focused on reducing product development costs or building a symbiotic relationship with the OSS communities [38]. The "when" concerns whether a reactive or proactive strategy is adopted. In the former, an organization adapts to existing and upcoming OSS communities without taking any initiatives. In the latter, an organization has a long-term agenda and adapts its OSS community engagements or create new communities accordingly.

2.2 Costs and Risks of Sharing Software as OSS

Deciding what should be contributed is a complex matter [159]. Even though the amount of a software that may be considered differentiating is often limited [134, 212], the risk of sharing differentiating functionality and sensitive Intellectual Property Rights (IPR), and as a consequence losing a competitive edge, is a recognized challenge in literature (e.g., [86, 87, 96, 212, 222, 231]). Instead of disqualifying a complete software artifact however, one approach may be to selectively reveal commodity or enabling parts while keeping differentiating parts

closed [86, 205, 219]. An alternative approach may be to "spinout" [222] disclose the software artifact under a restrictive copy-left license [219], e.g., the General Public Licence version 2 and 3, or the Affero GPL[2]. This is a common approach for commercial OSS organizations [186] using a dual-license approach to appropriate and capture value from customers [32]. Combinations can be found, e.g., where a core OSS project is permissively licensed, while certain extensions or improvements are licensed with more restrictive licenses [48], or kept proprietary [217].

A related challenge is determining when software should be contributed [231]. Several studies have attempted to model and identify an optimal timing [28, 82, 115]. Caulkins et al. [28] for example, identify costs related to the development and adapting the business model, along with the software quality as factors affecting when software should be released as OSS. Quality is in this case not just referred to the number of errors, but to the software's features and functionality. The shift in how quality changes can be compared to where in the commoditization process a software artifact is. I.e., the *"software artifact's value depreciation and how it moves between a differential to a commodity state, i.e., to what extent the artifact is considered to help distinguish the focalorganization'sproduct offering relative to its competitors"* [128]. As suggested by van der Linden et al. [212], a first step may be to open up the software in joint ventures or closed strategic alliances [128], while a third step may be to release it as OSS. Wnuk et al. [231] describe the challenge as a balance between losing a competitive edge and increased maintenance costs.

Where to contribute is a third challenge. When contributing the software artifact to an existing OSS community not governed by the organization itself [172], the organization needs to consider the requirements engineering process of the community [192]. In this context, the organization is a stakeholder among many and needs to consider potentially conflicting agendas from other stakeholders [130, 138, 159, 194]. If there is a miss-alignment between the organization's and the community's RE process, the organization may need to influence the development direction of the community [157]. If the organization lacks the influence needed, they need to consider the cost of gaining it [129]. An option is to release the software as an independent OSS project and build a new community around it, which may require significant investment as well [39, 109, 223].

2.3 The Contribution Acceptance Process Model

The Contribution Acceptance Process (CAP) model [128] was designed with the purpose to help organizations decide if a software artifact should be shared as OSS. A software artifact (e.g., features or projects) is valued according to its *business impact* (how much you profit from the component) and *control complexity* (how hard the technology and knowledge behind the artifact is to acquire and control). With the help of a series of questions provided, the software artifact could be

[2] https://opensource.org/licenses/alphabetical

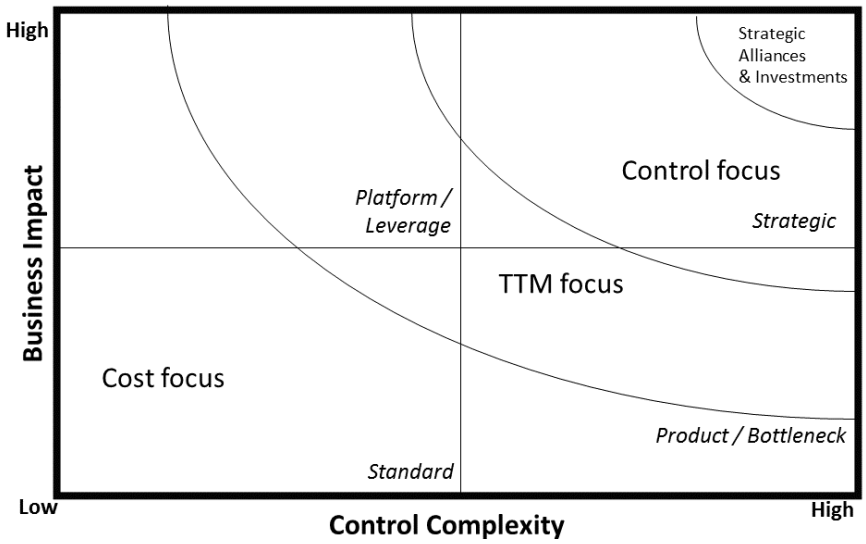


Figure 1: The Contribution Acceptance Process (CAP) model as presented in Linåker et al. [128]. Based on how a software artifact is valued in regards to its business impact and control complexity, a contribution strategy is elicited pending the artifact's placement on the grid.

ranked qualitatively and placed in a two-by-two matrix (see Fig. 2.3), with each cell representing a certain type of generic artifact with its own contribution strategy that is proposed for the artifact.

As an example, one of the four artifact types concerned strategic artifacts (upper right quadrant in Fig. 2.3) [128]. These artifacts have a high business impact and a high control complexity. They contain differentiating value and provides a competitive edge to the organization. Differentiating parts should be kept closed while enabling parts, should be contributed. The contribution is recommended to be made to a community where the organization has a high level of influence. If not possible, a new community may be created.

The contribution strategy could then be fine-tuned based on a series of objectives [128]. The more strategic an artifact is (higher business impact and control complexity) the faster the artifact (or enabling parts of it) should be contributed in order to establish the artifact as the standard solution. The organization could thereby avoid having to adapt to competing solutions and instead strive towards steering its competitors. For this reason, these kinds of artifacts should be prioritized before standard artifacts, which may be considered as standard knowledge.

An example of a strategic artifact included a multimedia framework that en-

abled differentiating camera functionality [128]. The framework was contributed, while the camera functionality could be kept closed. This gave Sony Mobile the advantage of not having to refactor or adapt to competing frameworks and still keep differentiating functionality closed.

The CAP model can be used both in a proactive and reactive approach [128]. In the proactive, it is used to map a series of software artifacts, e.g., features in the product planning process [111]. A cross-functional team may be assembled, including e.g., representatives from marketing, legal and product development. The team can iterate the mapping process comparing features against each other through consensus-seeking discussions. In the reactive approach, the CAP model is used in a similar manner but for making decisions in regard to incoming contribution requests from the organization's development teams.

2.4 Contribution Processes

A contribution process starts with an individual filing a contribution request, requesting to be allowed to contribute a certain software artifact to a certain community or create a new community. The request is then managed by an entity within the organization that has the mandate to decide on a contribution strategy.

In the case of Sony Mobile [128,151,157], an individual from the development organization fills in a contribution request answering a set of pre-defined questions. One of these questions concerns the size and complexity of the contribution [128].

- *Trivial contributions* include small changes such as bug fixes and minor improvements to existing OSS communities.
- *Medium contributions* include new features and larger architectural changes to existing communities.
- *Major contributions* include projects where a new community is to be established or software artifacts containing important IPR such as patents.

Trivial contribution requests are decided on by the closest manager, while medium and major contribution requests are processed by an Open Source Governance board [128]. The board has a cross-functional constellation with competencies covering aspects such as legal, user experience, product development, and product ownership [157]. Each medium and major contribution request is investigated further, which includes an IPR review rendering in a decision from the board on a relevant contribution strategy [128]. To ease bureaucracy, the board can develop general contribution strategies for specific communities allowing developers to, e.g., contribute minor and medium contributions to an OSS community without having to submit a request. This approach was applied to OSS communities where the OSS project was considered not providing a competitive edge to Sony Mobile, e.g, the two development-tools Jenkins and Gerrit [157].

Creating awareness and maintaining the contribution process can be a challenge in large organizations [231]. In Sony Mobile, the contribution process and Open Source Governance board are led and supervised by the Open Source Program Officer, a role that connects management, legal, IPR and software development functions around Sony Mobile's OSS operations [151]. Similar organizational and process setups have been reported on in the documents and guidelines published by the TODO-group[3], a foundation for organizations that has an established Open Source Programs Office. The program's office can be viewed as an organizational entity including the Open Source Program Officer and any further roles tied to an organization's OSS operations, e.g., concerning compliance and community management. In his more compliance-focused overview of OSS governance within an organization, Kemp [107] proposes the creation of an "OSS working party" and an "OSS compliance officer", which to some extent can be compared to an Open Source Programs Office and an Open Source Program Officer.

2.5 Summary

The benefits that may incentivize an organization to contribute its software as OSS are many [84, 93, 159, 197], as are the potential costs and risks that may remove or outweigh the benefits [131, 159]. To gain the expected benefits, an organization, therefore, needs to consider the contribution objectives and complexities relevant to their context, and make an informed decision on what they contribute, where, and when, in alignment with their business goals. Related work on how organizations can develop such strategies is limited [159], with some exceptions [217,231], including our previous work with Sony Mobile and the CAP-model [128]. The CoMn framework presented in this paper is created using a design science research approach [228] with three design cycles, where the first cycle is constituted by the already reported CAP-model [128]. The second and third design cycles reported in this paper cover a multiple-case study involving three case organizations (CaseOrg1-3). For a discussion on the relation between the CAP-model and the CoMn framework, see Section 6.3.

3 Research Design

This study uses a design science research approach, which aims to understand a real-world (design) problem, derive and answer knowledge questions, and design and validate an artifact that can be used by practitioners as a potential solution or treatment to the identified problem [3, 90, 228]. The steps mentioned can be divided into three phases: problem investigation, artifact design, and artifact validation [228]. As design science research is a creative process [90], these phases

[3] https://todogroup.org/

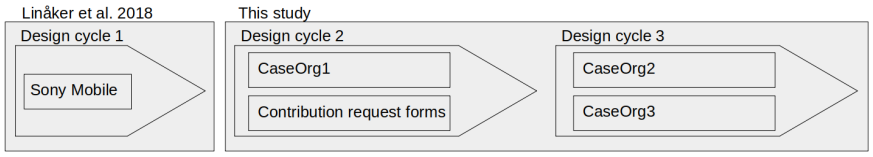


Figure 2: Three design cycles were used in the design of the artifact presented in this study. The first cycle is reported and presented in earlier work [128].

are carried out iteratively in design cycles. Knowledge gained from studying the problem in its context is used as input to design an artifact that might potentially treat the problem, which is then validated. After a cycle is completed, feedback from the artifact validation is used as input to a new design cycle. When validity is believed acceptable, the artifact can be transferred, implemented and evaluated on a problem instance in a real-world setting [79, 228].

The artifact presented in this study is the result of three such design cycles (see Fig. 2) using a multiple-case study [191] as suggested by van Aken [3]. The first design cycle has been reported on in an earlier study where an initial artifact was designed in collaboration with Sony Mobile [128]. In the second and third research cycles, reported on in this study, three more case organizations are studied. The unit of analysis in each case study is the case organizations', either implicit or explicit, contribution strategy decision process. Below we present each design cycle in detail, including a brief summary of the first and earlier reported cycle [128].

3.1 First Design Cycle

The *first* design cycle has been reported on and presented in an earlier study where an initial artifact referred to as a Contribution Acceptance Process (CAP) model was designed in collaboration with and in the context of Sony Mobile [128]. For a brief description of the CAP model, see Section 2.3.

As part of the validation of the CAP model, a workshop was arranged at the case organization referred to as CaseOrg1 in this study. The organization operates in the telecommunications domain and is reliant on OSS in order to develop and deliver its products and services (for further detail, see Section 4). The CAP model was applied to an internally developed infrastructure project at CaseOrg1. The workshop, along with two other case studies reported in the previous study [128], showed that the CAP model provided a good foundation for discussion. Feedback pointed out that the questions and scale used to value a software artifact in terms of business impact and control complexity, was found useful but in need of being tailored to the context where the CAP model is applied. A highlighted concern for future design cycles was not to avoid making the following versions of the CAP model more complex.

3.2 Second Design Cycle

In the *second* design cycle, a design decision was made based on learning outcomes from the first cycle to make subsequent artifact design more general. An organization should be able to elicit and tailor contribution strategy guidelines and a related contribution process based on the organization's specific context. Based on the business impact and control complexity dimensions of the CAP model, **RG1-2** (see Section 1) were defined for the second and third design cycle.

To address the two research questions, a questionnaire (see Appendix in Section 9) was created based constructs from the CAP model [128] and used in six semi-structured interviews with employees from different areas of the organization, see Table 1. Interviewees were selected in order to gain different and complementary perspectives on CaseOrg1's contribution strategy decision process. Each of the interviews was audio recorded and transcribed. An anonymized interview summary was presented to I1 and I2 to verify interpretations and clarify any misunderstanding. The transcripts were then coded with an inductive approach and audit trails maintained [191]. Using an iterative and refining coding process [187, 191], sentences and paragraphs from the interviews were given descriptive topic sen-

Table 1: List of interviewees from CaseOrg1-3.

ID	Case organization	Title	Employment
I1	CaseOrg1	Open Source Program Officer	3 years
I2	CaseOrg1	Community Manager	9 years
I3	CaseOrg1	Director of Software Development	9 years
I4	CaseOrg1	Director of Software Architecture	14 years
I5	CaseOrg1	Vice President - Standards	16 years
I6	CaseOrg1	Chief Software Architect	13 years
I7	CaseOrg2	Project Manager	1 year
I8	CaseOrg2	Senior Developer	4 years
I9	CaseOrg2	Junior Developer	1 year
I10	CaseOrg2	Department Manager	5 years
I11	CaseOrg2	Business Unit Manager	1 year
I12	CaseOrg3	External Consultant	3 years
I13	CaseOrg3	Chief Digital Officer	6 years
I14	CaseOrg3	Department Manager	6 years
I15	CaseOrg3	Product Owner	9 years
I16	CaseOrg3	Project Manager	2 years
I17	CaseOrg3	Community Manager	1 year
I18	CaseOrg3	Security Engineer	1 year
I19	CaseOrg3	Senior Developer	6 years
I20	CaseOrg3	Senior Developer	5 years

tences which then merged together in common codes and sorted under the two categories Contribution Objectives (COs) and Contribution Complexities (CCs), mapping to **RQ1** and **RQ2** respectively. This rendered in a first draft of the CoMn framework presented in this study, containing 16 COs and 12 CCs (see table 3).

In order for organizations to integrate their contribution strategy guidelines into a contribution process, *contribution request forms* were collected from seven different software-intensive organizations (see Table 2). A contribution request form contains questions about software artifact, and often guidelines for, and examples of what types of contributions the organization generally accepts. The form is often submitted by the engineer or team behind the concerned software artifact. The contribution request forms give an indication of what the organizations consider when deciding whether the software artifact should be contributed. Questions and statements from the forms were iteratively grouped and categorized in a table, adding one organization at a time. The final grouping and categorization are presented in Table 6 (see Appendix in Section 11) with example questions and statements. The questions were cross-mapped against the first draft of the framework.

Table 2: List of organizations from where contribution request forms has been gathered.

ID	Business	Market	Employees	Use of OSS
O1	Consumer electronics	Worldwide	4 000+	Infrastructure & Products
O2	Consumer electronics	Worldwide	100 000+	Infrastructure & Products
O3	Software products	Worldwide	30 000+	Infrastructure & Products
O4	Software products	Worldwide	100 000+	Infrastructure & Products
O5	Telecom	North America	1 000+	Infrastructure & Products
O6	Consumer electronics	Worldwide	1 000+	Infrastructure & Products
O7	Industry organization	North America	-	-

3.3 Third Design Cycle

To continue the design process beyond CaseOrg1, two new case organizations (CaseOrg2-3) were investigated in a *third* design cycle. CaseOrg2 develops and sells hardware devices with OSS-based embedded systems, while CaseOrg3 is

the Swedish Public Employment Service, a public sector agency in Sweden. For further detail, see Section 4.

To investigate the problem context within the organizations, five employees from CaseOrg2 and nine employees from CaseOrg3 were interviewed using semi-structured interviews. As in the former cycle, interviewees were selected in order to gain different and complementary perspectives on two organizations' contribution strategy decision process. The questionnaire used for CaseOrg1 was revised based on the findings from the second design cycle (see Appendix in Section 10). All interviews were audio recorded and transcribed. Anonymized interview summaries were presented to I7 and I8 from CaseOrg2 and I15 from CaseOrg3.

Using the coding schema from the second design cycle, transcripts from CaseOrg2 was first coded, which resulted in three new COs and one CC being added. In the following step, the new coding schema covering CaseOrg1-2 was applied to transcript from CaseOrg3, resulting in one new CO being added, six COs merged into three. Three new CC were added, of which two were former COs. Transcripts from all case organizations were then reiterated, and a final coding scheme assembled. This resulted in two COs being merged into one, two COs converted to CCs, and six CCs being merged into three. In table 3, the evolution and saturation of the COs and CCs throughout the coding process is presented.

Table 3: Evolution and saturation of COs and CCs throughout the coding process consisting of four steps. The first coding schema was based on transcripts from CaseOrg1. This was then applied and revised based on transcripts from CaseOrg2, and then CaseOrg3. The coding schema was then reiterated, resulting in the final set of COs and CCs.

	1. CaseOrg1	**2. CaseOrg2**	**3. CaseOrg3**	**4. Final coding**
COs (New)	16	3	1	0
COs (Merged)	0	0	6 →3	2 →1
COs (Removed)	0	0	2 →CC	2 →CC
COs (Total)	16	19	15	12
CCs (New)	12	1	3	2
CCs (Merged)	0	0	0	6 →3
CCs (Removed)	0	0	0	0
CCs (Total)	12	13	16	15

The mapping between the categorized questions and statements from the seven forms and the contribution objectives and complexities was then revised as presented in Table 6 (see Appendix in Section 11).

The contribution objectives and complexities of the revised CoMn framework were presented and discussed with I1 from CaseOrg1, I7 and I8 from CaseOrg2

and I15 from CaseOrg3. All interviews were audio recorded and transcribed. The interviewees were first asked about the correctness of the objectives and complexities that had been mapped to their organization. Interviewees were then asked if any objectives and complexities not mapped to them were found relevant for the organization. Lastly, the interviewees were asked about the general completeness, applicability, and usability of the framework, and whether something was redundant, missing, or could potentially be modified. The framework was then refined based on interview findings. No new contribution objectives or complexities were added.

4 The Case Organizations

Below we describe and provide context to each of the three case organizations studied. We provide a general description, along with a more in-depth overview of their contribution process as well as examples of OSS projects, which they have released, or are active contributors to.

4.1 CaseOrg1

General description: CaseOrg1 is a US-based media and technology company providing video, high-speed internet, smart home and voice services. They have 1000+ employees and develop their own software in order to enable and deliver their services to the customers. Having been passive consumers of OSS before 2006, they became active contributors starting in 2006. Since then, they have released a number of OSS projects and are active contributors in several others, as well as members of a number of OSS foundations.

Contribution process: Internally, they have an Open Source Programs Office set up to develop and manage e.g., contribution and compliance processes, and community management. Their contribution process starts by a developer filling out a contribution request form concerning a software artifact (e.g., feature or project). If the artifact regards a smaller contribution (e.g., a bug fix, or documentation), or a contribution to an OSS community deemed generally as non-competitive, approval can be gained online by the manager and the Open Source program office. For more significant components, the contribution request is managed by an OSS advisory board, which has representatives from legal, IPR, development and business functions within the organization. The board then gives a final decision on how to proceed. CaseOrg2 also has what they refer to as *sandbox approval*, which allows a project and identified contributors to be approved to contribute to a project without coming back for each patch. This governance set-up and contribution process is similar to that of Sony Mobile [128].

Example OSS projects: One example concerns an internally developed Linux distribution that is embedded in hardware devices shipped to customers, which

enables the delivery of CaseOrg1's consumer-oriented services. The software was initially developed to replace a proprietary option and was later released as OSS under the governance of an industry consortium. Another example concerns an infrastructure project, which enables the delivery of services related to secure and reliant internet-traffic to business-oriented customers. The project was released under the governance of a neutral community with an existing OSS foundation. A third example regards a DNS-as-a-Service project originally developed to increase internal operational efficiency. The decision process is thoroughly reported on in previous work [128].

4.2 CaseOrg2

General description: CaseOrg2 is a European-based hardware electronics manufacturer serving both business and private customers. They have 1000+ employees and develop their own software and orchestrate a software ecosystem [100] in order to enable and deliver their services to the customers. This study has focused on its Tools department which develops and maintains development tools and infrastructure projects used by the organization's product development teams. A majority of the tools and infrastructure projects are OSS-based with active engagement from the Tools department in their respective communities. The active engagement includes continuous contributions of features and plugins to existing OSS communities as well as the release of new OSS projects.

Contribution process: CaseOrg2 does not have a dedicated Open Source Programs Office. The organization has two contribution processes set up, one for their product development teams, and one for the Tools department. The latter is less strict than the former as CaseOrg2 generally considers the projects developed within the Tools department to provide a limited competitive edge to the organization. The contribution process used within the Tools department is initialized by a developer filling out a contribution request form concerning a software artifact (e.g., feature or project). If the contribution request is intended for an existing community, the request is managed by the department manager. If the contribution request concerns the creation of a new OSS community, the request is managed by the business unit manager. If deemed necessary the concerned manager will consult relevant functions within the organization such as Legal and IPR departments.

Example OSS projects: The Tools department have contributed and maintains a number of plugins to the Jenkins and Gerrit OSS projects. Jenkins is a build-server and Gerrit is a code-review tool, both used in CaseOrg2's continuous integration tool-chain. The Tools department has recently also started to create new OSS projects that fall outside existing communities. One such project was developed internally to allow for the creation and use of shorter URLs to internal resources. These are maintained as standalone projects on CaseOrg2's Github[4] page.

[4]https://github.com/

4.3 CaseOrg3

General description: The Swedish Public Employment Service[5] makes up the third case organization. They are non-anonymous but are referenced to as CaseOrg3 for consistency. CaseOrg3 is a public sector agency in Sweden with the main goal to facilitate and enable matching between job-seekers and employers on the Swedish labor market. The organization has 10 000+ employees of which 600 are employed within their IT division. The focus of this study is on a department within the IT division which aims to create a platform[6] on which private actors can build complementary products and services for matching job-seekers and employers. The platform, consisting of OSS projects and Open Data sources, is intended to help CaseOrg3 to move from the role of being a service provider to become a service enabler and help the platform's ecosystem members to collaborate and co-create, potentially resulting in accelerated innovation and a more efficient job-matching on the Swedish labor market.

Contribution process: CaseOrg3 does not have a dedicated Open Source Programs Office or contribution process in place. The studied department prioritize the release of internal projects based on what they believe is of most value to the platform's ecosystem of private actors.

Example OSS projects: The studied department has released a number of OSS projects consisting of small components that can integrate into existing applications, stand-alone end-user applications, and developer-focused tools and utilities[7]. One project is a video-conference-tool used internally at CaseOrg3 to facilitate remote interviews for employers and job-seekers. Another example concerns a search engine that can be used to access and search among job listings in CaseOrg3's Open Data sources with job ads.

5 Contribution Management Framework

In this section, the Contribution Management (CoMn) framework is presented based on a multiple-case study of three case organizations. The purpose of the framework is to enable software-intensive organizations to align its business goals and contribution strategies by supporting the organization in creating contribution strategy guidelines and related contribution processes for internally developed software artifacts. The framework consists of a series of contribution objectives (COs) and complexities (CCs), which an organization may consider and weigh against each other when deciding on a contribution strategy for a certain software artifact. A **contribution objective** is defined as *a purpose for contributing a software artifact, motivated by a monetary or non-monetary benefit that is enabled*

[5] https://arbetsformedlingen.se/
[6] https://jobtechdev.se/
[7] https://jobtechdev.se/doc/samples/

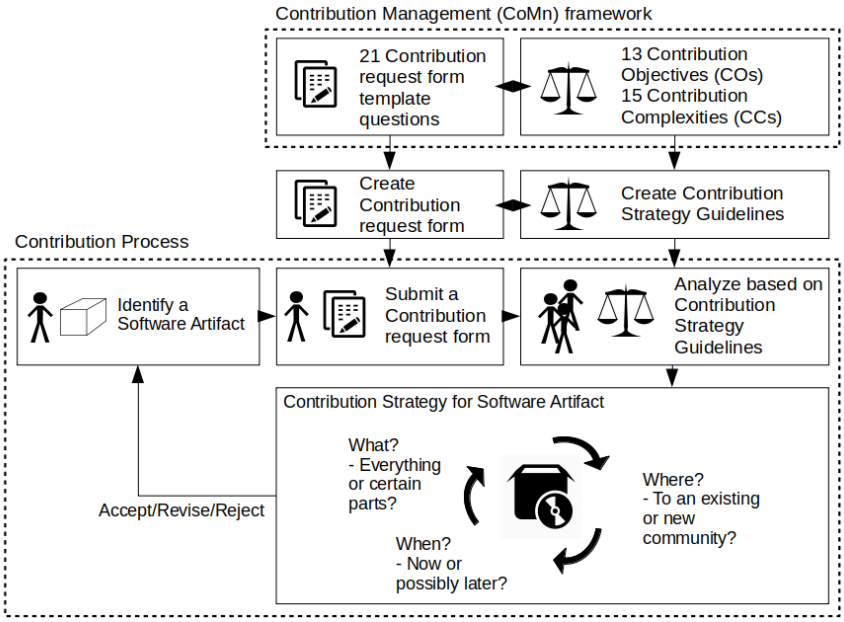


Figure 3: Overview of a contribution process, where an individual wishing to contribute a software artifact fills out a contribution request form, which is then analyzed based on an organization's contribution strategy guidelines, after which a contribution strategy is decided on. The CoMn framework can help an organization implement the contribution strategy guidelines as well as the related contribution request form.

or resulted directly or indirectly as a consequence the contribution. A **contribution complexity** is defined as *an aspect of or related to a software artifact that may complicate the contribution of the artifact, or imply a cost or risk as a result thereof, either directly or indirectly*.

Not all contribution objectives and complexities may be relevant for the organization in general or in certain cases. To help guide decision-makers and those making contribution requests, contribution strategy guidelines can be developed based on the objectives and complexities that are identified as generally relevant for the organization. The identified objectives and complexities can then be described in the context of the focal organization and relevant questions asked in a contribution request form when setting up a contribution process within the organization. An individual intending to file a request is then enabled to beforehand understand the rationale used by the decision makers when deciding on a contribution strategy, and thereby to provide motivated arguments why the request should be accepted.

To also support the request step of the contribution process as visualized in Fig. 3, the CoMn framework also provides a series of template questions that map to the contribution objectives and complexities. These template questions can, therefore, be used to design contribution request forms specific for an organization's contribution strategy guidelines.

The contribution objectives of the CoMn framework are presented, each with a number of (potential) key benefits, in Table 4, while the contribution complexities are presented, each presented with a number of key concerns, in Table 5. They are further explained below, followed by the mapping of contribution request questions.

5.1 Contribution Objectives

Below each of the contribution objectives is described separately using quotes from interviewees of the case organizations where the objective was identified.

Table 4: Overview of the contribution objectives, related key benefits, and in which case organization they were identified.

ID	Contribution objective	Key benefits	Case Org
CO1	Build a software ecosystem	• Creation of more and better market-oriented solutions • Open innovation for a whole market	2,3
CO2	Prove skill and influence	• Improved trust among customers and partners	1,2

CO3	Create price pressure	• Lower subscription costs of procured products and services • Lower prices on tenders	1,3
CO4	Standardize solution	• Force competitors to adapt to the contributed solution (CaseOrg1-2) • Steer market and community according to internal agenda (CaseOrg1-2) • Improve competition and enable more value-adding development on market (CaseOrg3)	1,2,3
CO5	Increase transparency	• Improved trust among customers and partners	1,3
CO6	Improve employer branding	• Recruitment of talented employees • Retention of existing employees	1,2,3
CO7	Improve partner collaboration	• Increased value of software ecosystem	2
CO8	Outsource infrastructure operation	• Internal focus on activities related to core business	1
CO9	Open up innovation process	• Accelerated innovation process • Creation of more and better market-oriented solutions	1,2,3
CO10	Extend workforce	• External contributions including features, improvements and bug corrections • Accelerated development • Improved quality assurance • Lower maintenance cost • Focus on more value-adding development	1,2,3
CO11	Collect data	• Improved machine learning and artificial intelligence algorithms. • Creation of solutions based on Open Data sources.	1,3
CO12	Be a good open source citizen	• Idealistic satisfaction among employees.	1,2,3

CO1 - Build a software ecosystem: An OSS project can by itself be seen as a technological platform and the related community as the software ecosystem which is underpinned by the platform [100]. However, on a higher abstraction level, multiple OSS projects may also be seen as a technological platform [72]

and the related overlapping communities as a single software ecosystem [105]. In CaseOrg2, a long-term goal of the Tools-department is to standardize all tools that are used in the industry as OSS projects and build a larger software ecosystem around these. From a public sector perspective, CaseOrg3 is also aiming to create a larger software ecosystem by contributing internal software artifacts as new OSS projects along with Open Data sources. Based on the platform, constituted by the OSS projects and Open Data sources, private organizations and firms on the labor market can create new and *"hopefully better market-adapted solutions"* (I13). I17 adds *"I think there is an interest and a need for everybody within the sector to collaborate and to help each other and to make sure that the right individuals land the right opportunities. So what we are trying to gain is literally a collaboration [...] with others to provide better services to society"*.

CO2 - Prove skill and influence: Being an active contributor in a community can help to build a reputation of an organization as technically skilled and influential in the community, which can be valuable attributes to communicate both towards partners and customers [159, 213]. I1 gives the example *"It helps us from a brand-building perspective because the project is seen as being on the cutting edge of AI"*. In another example, *"We basically realized that we were so dependent on [OSS project] and that it was the selling point of our product, so we needed to demonstrate for customers that we were one of the core contributors. It was a selling point in the marketing brochure"*. I7 adds how being an active contributor in a community can help to promote CaseOrg2 in new channels and build up new partnerships.

CO3 - Create price pressure: Releasing a project as OSS can be a way to put price pressure on third-party vendors and suppliers of proprietary solutions. As a reaction to an expensive proprietary infrastructure solution, CaseOrg1 developed an internal replacement and open sourced it as *"a competitive advantage in the negotiations against [the supplier of the proprietary solution]"* (I4) in order to *"drive the cost of doing business down"* (I5). From a public sector perspective, by releasing software as OSS, and requiring tenders to build on it, competition in the tender process is improved as smaller actors enabled to participate in a process *"otherwise set-up to benefit large firms"* (I12).

CO4 - Standardize solution: By having an OSS project or contribution adopted and accepted as a standard solution, an organization can potentially replace existing proprietary and community solutions [86, 98, 134, 219]. As an example, CaseOrg1 developed and open sourced a project to replace proprietary solution as *"a way to get away from a commercial vendor and open up the access to the data collected by the vendor, but also to be able to influence and control [the project's] direction"* (I6). As further explained by I1, *"by putting it out there you can influence it and thereby where the market is heading"*, potentially steering the competition in the direction most beneficial to the firm. In another example, CaseOrg2 wanted to replace an existing plugin related to an OSS project they are dependent on. If successful, it would *"mean less of a burden when upgrading to*

new releases" and *"not having to maintain an internal fork"* (I8). From a public sector perspective, CaseOrg3 views the benefit as *"improving competition"* (I13) and *"allowing more to join the market"* (I15). I15 continues, *"I think we are raising the bar for the entire market, allowing everyone to level up and stop focus on base infrastructure"*.

CO5 - Increase transparency: By being transparent in how an organization performs certain actions, they can build trust among customers and potentially avoid allegations of wrongful doing. As an example, CaseOrg1 developed and open sourced a tool to measure the speed of their services. By keeping the project open, the organization could be transparent in how they performed the measurement. From a public sector perspective, CaseOrg3's aim is in a similar manner create transparency towards the public and thereby earn their trust, *"this is what you get for your tax money"* as put by I13.

CO6 - Improve employer branding: Working with OSS projects can help an organization to both retain existing and attract new engineers [159]. The communities make up talent pools where organizations can access a skilled and specialized workforce. which would otherwise be hard to find, as emphasized by I10. Besides being allowed to engage with an external community with the intrinsic rewards that follow, highlighted by I19 as *"the fun parts"*, working with OSS offers engineers the opportunity build their own transparent portfolios. As stated by I6, *"[the engineers] view it as a core part of their career development and in a way, they can kind of be recognized for that in a very different way than they could if they were simply working on internal or commercial software"*. I8 adds how it helps in *"building [the engineers'] CVs"* by allowing *"the work they do to become transparent to the outside world"*. I5 adds *"It is important to people and I think a positive employment characterization that you get to engage with open source communities and that the company does release something as open source projects. I think that is a big selling point that people are looking for in whom they want to work for as an engineer"*.

CO7 - Improve partner collaboration: Sharing e.g., tools and infrastructure projects as OSS could potentially act as a complement and improve collaboration between CaseOrg2 and other actors within CaseOrg2's software ecosystem that is connected to its core-business of embedded devices. As put by I7, *"It would be easier to collaborate around the same stack. With some partners, we already have a collaboration, while others only use our services. Would we, however, open source more of our tools they could start to contribute back... Also, by offering boilerplates to our partners we help them to do the right thing from start"*. I8 adds, *"I think, would [CaseOrg2] be more open with its tools and libraries, this would be appreciated by, and a way to come closer to our third-party developers"*.

CO8 - Outsource infrastructure operation: Sharing software as OSS can be seen as a way to not just outsource the development of a software project [2], but also the operation of it. As explained by I5, *"We can get to a point where we can actually outsource the operations of this thing that we actually built to*

somebody else, via an OSS path, which then frees up this team to do other stuff". I5 continues, *"So there's an interesting option which is like, well we need it so we are going to build it, run it in [Amazon Web Services] ourselves, but then we will open source it, almost in the hopes that Amazon will steal it and run their own service of it, and then we will just use theirs because they can operate it more cheaply than we can"*.

CO9 - Open up innovation process: Collaboration with OSS communities can be seen as a case of Open Innovation [159], allowing an organization to extend its R&D and innovation capabilities and initiate new collaborations in an open and transparent way. I6 describes it as *"Open source is in many cases about doing external R&D in a way. You are leveraging external groups to do things, and it is often the case that many companies don't really fund medium to longer term R&D in-house anymore, there's really [not many] Bell Labs to speak of. So you have communities that are doing that and you have engineers from companies that might work on those things, and it is a lot easier with those external projects, and think you get the network effect of so many more participants"*. CaseOrg3 sees the open collaboration as an opportunity to *"open up the requirement engineering process"* (I15) and create *"faster feedback-loops, and to become more reactive and compatible to the needs of the market"* (I12).

CO10 - Extend workforce: Using and combining the development resources of an OSS community with an organization's internal, software development may potentially be accelerated and extended as e.g., features are implemented and bugs corrected [157]. As explained by I6, *"Now you have really everyone across the industry focused on largely open source, and as a result of that you are getting a lot of internal development teams that are contributing to the code, finding issues, not just depending on an external community to do it, and as a result of that, you almost get this acceleration of innovation on that platform. It supercharges [the platform] in a way and you are getting, maybe it is us and five other actors, maybe even competitors, and we are all contributing back to things that we are finding. It is a better product for everybody. So I think that accelerates the development process"*. The shared development and maintenance may further imply that resources are freed up and can focus on more value-adding activities. I6 describes it as *"the more that you can push things down to commodity, the easier it becomes, and cheaper, and then you can keep focusing more up the stack. Keep focusing on the next new feature"*. In the case of CaseOrg2, the external workforce provides an important leverage as the organization generally have *"limitations in terms of time and resources"* (I7). From a public sector perspective, I13 views OSS as a way to generate a *"higher value in return for the tax money invested"* compared to the closed option.

CO11 - Collect data: CaseOrg3 views some OSS as an enabler for others to use and contribute to their Open Data sources. By sharing e.g., algorithms as OSS, others can generate data which can then be used to improve artificial intelligence and machine learning initiatives. I15 exemplifies how the gathering of job-ads

can help create an automated review-function which could classify if an add is discriminatory or not. At CaseOrg1, I1 explains, *"The Machine learning team has often come to us saying, I want to open source this algorithm because I wanna learn how it grows, and we alone don't have enough data to feed and we need the world to feed it data"*.

CO12 - Be a good open source citizen: Contributing to an OSS project may for some be ideologically motivated [98] and seen as *"the right thing to do"* according to I10 who wishes to enforce an *"open per default"* policy. I5 explains that CaseOrg1 *"is much better of contributing that back to the community"*. I5 continues, *"I think it is from one point of view the right thing to do, because you are taking advantage of this set of code, and so you should, while you are taking advantage of it, you should also be contributing back [from] a good citizen point of view"*. I5 further adds that a reason *"may be that the developers are pro-OSS and so they are just like, this isn't secret sauce, so just sort philosophically we want to contribute this to the commons because maybe someone will take advantage of it. There's a good citizen aspect to it"*.

5.2 Contribution Complexities

Below each of the contribution complexities is described separately using quotes from interviewees of the case organizations where the complexity was identified.

Table 5: Overview of the contribution complexities, related key concerns, and in which case organization they were identified.

ID	Contribution complexity	Key concerns	Case Org
CC1	Ethical use	• Risk of creating negative exposure, hurting brand and public trust.	3
CC2	Impact on value proposition	• Risk for misalignment between internal and external agendas. • Risk for loosing control of development.	1,2
CC3	Impact on internal operations	• Risk for misalignment between internal and external agendas. • Risk for loosing control of development.	1,2
CC4	Differentiating functionality	• Risk for giving losing competitive edge (CaseOrg1-2). • Risk of hurting the business models of existing actors on market (CaseOrg3).	1,2,3

CC5	Commoditization	• Risk for giving away competitive edge or differentiating functionality too early (CaseOrg1-2). • Risk of an alternative solution being accepted before ones' own (CaseOrg1-2). • Risk of hurting the business models of existing actors on market (CaseOrg3).	1,2,3
CC6	Sensitive IPRs	• Risk of giving away patented, patentable or in other ways sensitive IPR.	1,2
CC7	Security threats	• Risk of exposing security-related vulnerabilities.	1,2,3
CC8	Budget and resource constraints	• Cost for preparing, contributing and maintaining software artifact. • Risk for hidden and escalating costs	1,2,3
CC9	Modularity and architecture	• Technical feasibility to modularize and abstract software artifact.	1,2,3
CC10	Code alignment	• Misalignment between internal and external development may prevent or complicate future contributions. • Cost of maintaining internal fork.	1,2
CC11	External interest	• Risk of contribution not being accepted or community not being established.	1,2,3
CC12	Influence in community	• Low level of influence in the community may prevent or complicate the contribution of a software artifact.	1,2,3
CC13	Community health	• Contribution of the software artifact may help to improve the health of a community.	1,2,3
CC14	Substitutes	• Unnecessary cost of contributing if existing alternatives are considered as good or better.	1,2,3
CC15	License compliance	• Risk for violating license conditions if software artifact is not contributed.	1,2,3

CC1 - Ethical use: By releasing a software artifact as OSS, anyone is allowed to use it under the same conditions provided by the OSS license. Hence, a risk is that the software may be used for purposes not originally intended, which can *"include cases where we would not be very proud of as a public agency"* (I18). CaseOrg3 is, therefore, as explained by I17, *"trying to make sure that the code and the tools that [CaseOrg3] work on are going to be ethically used, and not*

misused". I18 raises the concern, *"how far does the responsibility of CaseOrg3 as a public agency stretch?"*.

CC2 - Impact on the value proposition: Software artifacts that have a close connection to an organization's value proposition and its revenue stream may warrant special attention in order to determine any potential costs or negative risks that may be implied by contributing the artifact. In the case of CaseOrg2, the department studied in this paper focuses on infrastructure and tools-projects which is used by CaseOrgs2's product development teams. The software developed and maintained by the department can hence be considered to have an indirect connection to CaseOrg2's value proposition and revenue streams, or as put by I7, *"not even close to core business"*. This provides the department with a *"much less restrictive process than what applies for core business"* (I7). CaseOrg1 develops and maintains similar types of projects, but also those that have a stronger connection to the organization's value proposition and revenue streams. For example, one project that they have released is a Linux-based operating system embedded in hardware devices which enables the organization the deliver its service offerings to its customers. Another project released by the organization makes up a pivotal part of their infrastructure, also enabling the delivery of its service offerings, but also the possibility to offer the infrastructure as a service to business customers, including competitors. In both examples, the projects are considered as commodity but of strategic importance why the organization needs to be *"actively involved and be able to steer the direction and make sure that as that project evolves that it continues to meet our needs and to evolve in a certain way"* (I5).

CC3 - Impact on internal operations: Even though a software artifact may have a loose connection to an organization's value proposition and revenue stream, it may still be of strategic importance internally. As in the case of CaseOrg2, *"We are extremely dependent on [OSS project] as we have built our whole continuous integration tool-chain on it, same goes for [OSS project]. Hence, we need to be active so that we can affect the direction on the projects, otherwise, it could become extremely costly for us. Better tools enable us to make better products"* (I8). I1 phrases it as, *"Which projects are we actively using as a company that we are so dependent upon for our success that we need to be at the table? You can do a survey on the company and ask, what open source infrastructure are you using, and how critical is it to your success? And how are you engaging today? And what is missing in your engagement?"*. I4 adds, *"We do need to influence [the OSS project's] roadmap because as the project continues to evolve, and we are such a huge user of it, we still need to drive its roadmap quite a bit to be able to get maximum value from it"*.

CC4 - Differentiating functionality: Giving away software artifacts that provide product differentiation or other types of competitive edge is a fear among both CaseOrg1 and CaseOrg2. However, I3 explains, *"when I look at what we've approved so far, practically 93 percent of what's coming in front of the committee we approve. The 7 percent has been the things that are core to our business, or some-*

thing that is a competitive advantage". This may be a consequence of CaseOrg1's work to actively encourage its developers to separate between differentiating and commodity functionality as explained by I6, *"These things are really basic functionality, that is ok, we are going to share that, that is part of the core platform. But there are maybe some parts where you can on a modular basis extend it or do integrations with internal systems in order to not give away the secret sauce, the differentiation"*. In the case of CaseOrg2, the department studied in this paper works actively to avoid any connections and dependencies to the organization's products. A bigger concern relates to the risk of giving away process-based competitive edge, as explained by I7 *"Of course we have tools that can provide us with a competitive edge"*. Faster development pace and better quality assurance are two areas highlighted by I9. According to I7, *"these components are often so small and specific to [CaseOrg2] that they are not worth to modularize"*. Concerning CaseOrg3, they prioritize releasing software artifacts that have the potentially highest differentiating value for actors on the market. However, as expressed by I15, *"It is a balancing act. There will be cases where we will open up stuff that some make a living on. Actors will have to innovate their business models and adapt. Our aim is to improve competition, not hurt it"*.

CC5 - Commoditization: The timing of when to release something as OSS can be a complex issue. As I1 highlights, *"There is a trade-off between a competitive edge and burdon"*. I.e., while keeping the software closed may provide a competitive advantage, it may cost in terms of maintenance and support. Another facet is the risk a competing solution being released and accepted before ones' own which I1 describes as a *"waiting game"*. I1 continues, *"[some] may want to say - I want to hold on to this feature longer - because they don't want to make it a level playing field for everybody, or they may take certain features and say, - I think we need to get it out there as fast as possible, because we heard that competitor X is working on putting this into the pipeline... So I think it could be both ways, there could be some features where we say, No, let us hold on to this until we get a first-mover advantage on the market using it, then once we have a significant advantage, then it is ok to commoditize it and put it out there"*. By releasing it before, an organization can avoid ending up with an internal fork and with the option of having to adapt to someone else's solution. Furthermore, the organization would have a bigger influence on the direction of the solution's development.

CaseOrg2 also sees value in being quick to release and contribute software before competing solutions are adopted. This is a discussion they have iteratively in their community engagements, especially for the Gerrit OSS community, where it is important for the organization to steer and influence the roadmap and long term planning of the community. I1 further adds, *"many companies do that, where they say, this user interface or this management console, or this testing that we have done, we don't want others to benefit from... until they see that it is becoming commoditized, or that someone else is thinking of contributing it, that is when they say, yes, maybe we should, before they do, because then their standard will become*

the standard, and we will be cut out, and so I think they play the waiting game for as long as they can". CaseOrg3 also considers the commoditization process as an important aspect, but as highlighted in CC4 they strive to share software artifacts that are differentiating in order to help businesses focus on more value-adding activities.

CC6 - Sensitive IPRs: Sensitive IPs or patents is not limited to functionality that provides a competitive edge in terms of a product or process (see CO4). For CaseOrg1, patents can be viewed as a competitive edge even in cases where they do not cover functionality that is actively used. As I1 explains, *"We live in a very litigious environment, which means, as a company we are in an industry where there are lots of lawsuits against each other, so we use our patent portfolio in a defensive way. So there is a drive to build a patent portfolio which is why the patent office encourages people to seek patents, because the bigger and better the patent portfolio, the more we can defend ourselves"*. Hence, a common question that is asked in CaseOrg1 is *"can and should we patent this?"* (I1). I1 also adds that in other organizations certain patents may provide license revenue, which should also be weighed against the potential value of sharing the patent as OSS. In the case of CaseOrg2, as they are orchestrating a larger software ecosystem around its products, a question they ask is if there are any patents or IPRs that belong to one of their partners.

CC7 - Security threats: There is a generally open attitude regarding OSS and the security aspect among all three CaseOrgs. At CaseOrg1 they ask *"does this expose the company to any security threats? and they usually answer no"* (I1). At CaseOrg2, *"the security department has a positive view on open source as you get more eyes on the code"* (I11). However, they still have a careful review process in place as they do not wish to expose any potential back-doors that could lead to theorganization'shardware-based products. At CaseOrg3, I18 highlights that *"it is the data that is considered valuable and needs protection"* which is rather done by *"proper key management"* than keeping any related software closed.

CC8 - Budget and resource constraints: Contributing a software artifact always comes with a cost. Abstracting, modularizing and generalizing an artifact may prove an expensive effort compared to projects that are developed with the intention from the start that it will be contributed *"where you often can take it as-is and move it into the community"* (I7). As put by I8, *"if we build something that is tailored to and dependent on internal infrastructure, it is most often no idea to contribute it. In most cases, we can modularize and generalize it, but the cost can get really high, so it is a matter of if it is worth it or not"*. I9 further mentions that there *"is a lot of hidden costs"* that are connected to the contribution process, such as *"scrubbing"* or cleaning the source code and its version history of sensitive or unnecessary comments and information. Other costs include going through the contribution process of the related community, or creating a community if the software artifact is released independently of any existing community. In the latter case, much more resources are required long-term both in terms of managing the

community, but also maintaining and leading the software development within the community. As highlighted by I10, *"we prefer contributing to existing communities because it is expensive taking on the role of a maintainer in a larger project"*. With this background, it is important to also ask *"who is going to be the owner of this repo, and have you allocated the time to maintain it, and has your boss approved your time budget?"* (I1). CaseOrg3 recognizes the costs implied by sharing a software artifact as OSS as a concern, but sees it from a long-term perspective. I15 asks, *"What is the need of the labor market? If there is a potentially positive outcome, then I think we are prepared to spend the money needed"*.

CC9 - Modularity and architecture: In certain cases, it may be that the software artifact is too embedded in internal infrastructure, which makes it infeasible to modularize the artifact, as hinted by all CaseOrgs. I1 stresses that *"companies should start projects with the goal that it will be open one day, then they could architect it right from the start with generality and modules that can be contributed. Because one of the things that I am learning is that everything will one day be open, it is [only] a matter of time. If we start developing with open in mind then we have lots of more options"*. Specifically for CaseOrg1 and CaseOrg2, if it is found that certain parts of the software artifact for different reasons may not be contributable, an option is to modularize and abstract these parts so that the rest, e.g., a platform may be contributed. I3 describes how the Legal department at CaseOrg1 can be *"very hardline if it is clearly something they feel should not go out"* but that *"there have been a few cases where they have said, ok, this component, if you can carve it out from this, it is ok"*. I1 further adds, *"We cannot wholesale say no, the whole thing cannot be done. Are there pieces that we can put out, so we can still learn, whilst still maintaining the bigger picture, which we don't want to reveal?"*. The potential to *"carve out"* sensitive pieces of the software artifact needs to be considered in relation to the expected benefits as well as other complexities as mentioned in this framework, e.g., regarding cost and security-related complexities. An option may be to contribute the *"design document or the blueprint of the project itself so someone else can create it"*, i.e., documentation that in some way captures the knowledge from the underlying software artifact.

CC10 - Code alignment: By not contributing internally developed code that relates to a project, a misalignment between the internally and externally developed software may arise. A negative consequence may be that the organization unintentionally ends up on a fork that grows with time and creates unnecessary maintenance and patch-work. I8 explains it as, *"If you don't share, the community may take off in another direction. Then you will stagnate and come to a place that will be very hard to get back from"*.

CC11 - External interest: To contribute a software artifact to an existing OSS community, or to create a new community around it, there needs to be an external interest for the artifact. I3 describes it as *"filling in a gap"*. I5 asks the question, *"Is there a general purpose part of this that I can see multiple teams inside my company taking advantage of?"*. External interest should, therefore,

be investigated before proceeding with any contribution. In one example where CaseOrg1 ended up creating a new community, I6 explained, *"everyone needs to do it, and it is not really great agreement over what the right methodologies are"*. For *"existing communities that have been around for long you know, or at least get an indication, if it is going to be an uptake or not [of the contribution]"* (I7). Analyzing stakeholders' agendas within a community can further help. *"What's their strategy, whom do they have playing, what are they trying to get done, what chess moves are they making, and if so, which then informs what we contribute, when we contribute, and how strongly we need to be present"* (I1).

CC12 - Influence in community: The level of influence an organization has (or needs) on the RE process within an OSS community is a multifaceted complexity. If the external interest within a community is weak, or if the community is heading in a different direction, it may be important to have influence in order to create traction and approval for the contribution, but also to be able to steer its development if it is accepted [129]. I7 highlights that *"It is important for [CaseOrg2] to pick up a leading role in certain communities that we value as strategically important and as a potential way to get an edge against competitors in those communities"*. I1 explains it as, *"Most organizations start out as I'm a big consumer of it, and then here's where I need to change it, to fit in to my needs, and then almost at the same time it becomes, Oh, gosh, I'm so dependent upon it and I cannot afford to have it fail or change in a way that doesn't fit my needs. And then it becomes a realization, I need to influence it"*. If e.g., the current level of influence is deemed not high enough to be able to contribute and steer the software artifact, one option is to create a new community where the community's governance structure can be tailored based on the focal organization's needs. I1 explains how CaseOrg1 reasoned about the creation of a separate community behind an internally developed Linux distribution, *"We could have contributed that code to the Linux Foundation or the Apache Software Foundation and have them make it a broader project. But we ended up creating our own foundation and collected all the [relevant stakeholders] together to contribute towards this platform that we created. In a way, it is the right way because we wanted to make sure that it served the need of our [main customers] and it didn't get too broad and become an embedded system that everyone uses... So we want to maintain influence and be in a controlling position, so we are one of three members of the steering committee"*.

CC13 - Community health: Contributing to and engaging actively with a community is one way to support its health, i.e., ability to survive throughout time [214], which is an important aspect if an organization is dependent on a specific OSS project [129]. I1 explain CaseOrg1's approach as *"We care about the health of the project, will it die because there are not enough contributors maintaining it? We cannot let that project die because we are using it and we would have to swap out the code and go to something else"*. I1 continues, *"We prefer to use something that is already there, and not reinvents the wheel, but if it is not going to be healthy, then we would want to be there just to create a vibrant*

community, not for influence, but for health". Also, I10 views the health of the OSS community and its project as important why I10 looks down on *"parasitic behavior"* from actors in a community.

CC14 - Substitutes: If there are existing alternatives to the software artifact available, it may be questioned why the artifact should even be considered. In CaseOrg1, according to I3, *"One of the questions we ask in the open source contribution form is, is there an existing project that does the same thing or is this one something unique and new? We've had people that, say, I wrote an HTTP client, and I want to open source it, that question goes to that, hey, there are plenty HTTP clients, why are you writing another one?"*. However, sometimes it may be motivated if there is a strategic intent to replace an existing solution in a community or an industry standard. In CaseOrg2, interviewees emphasized that this is an aspect that should be considered in an earlier stage, long before any contribution request would be made. *"We definitely prefer to reuse what's out there and not reinvent the wheel if not necessary... We cannot be the only ones with this problem"* (I8). CaseOrg3 exemplifies how they chose to replace their internally developed search engine with the OSS option Elasticsearch[8] as it provided more flexibility.

CC15 - License compliance: In cases where the software extends or integrates with an OSS project, the OSS license of that project needs to be reviewed and respected. Copyleft and restrictive licenses may require that the artifact is contributed. I7 highlights, *"we must be compliant with the licenses we have in our source code"*.

5.3 Questions linked to Objectives and Complexities

In Table 6 in Appendix in Section 11, categorized questions elicited from contribution request forms of seven software organizations are presented. Each category is linked to related contribution objectives and complexities as presented in Table 4 and 5 respectively. These categories and questions may be viewed as a template for how an organization can adopt the contribution objectives and complexities of the CoMn framework and adapt contribution request forms to be used in the organization's contribution process. An individual intending to file a request is then enabled to understand the rationale used by the decision makers when deciding on a contribution strategy, and thereby to provide motivated arguments why the request should be accepted.

Questions Q1-15 are focused on describing the contribution (Q1-6), the motive of the contribution (Q7), potential risks(Q8-13), and costs (Q14-15). These questions, therefore, help to answer the *"what"* of a contribution strategy according to the earlier definition. Questions Q16-21 relate to the decision of *"where"* the contribution should go, either to an existing community or if a new community should be established. None of the categories and questions elicited could be linked to the aspect of *"when"*.

[8] https://www.elastic.co/products/elasticsearch

An organization should keep in mind that these questions have not been rephrased from their original contribution request form. This may imply a loss of contextual information and a need for the questions to be rephrased and put into the organization's own context. A concern could also be that some questions may be too complicated for an engineer to answer, e.g., estimating the costs of the contribution. An organization should further be careful with the amount of questions included in their contribution request form as this may potentially complicate the process, prohibiting contributions as an effect.

6 Discussion

In this section we first discuss the contribution objectives and complexities respectively in relation to literature and in the contexts they were observed among the case organizations. This is followed by a discussion on similarities and differences between the CoMn framework and the CAP model which the CoMn framework relates to.

6.1 Contribution Objectives

The expected benefits from sharing a software artifact as OSS is reflected in the CoMn framework by the contribution objectives and related key benefits (see Table 4). Objectives can be considered to cover both the idealistic, survival and commercial motivators highlighted by Jansen et al. [98]. Supportive findings can to a large extent also be found in literature if compared to Section 2.1. For example, the idealistically motivated objective CO12 - *Be a good open source citizen*, implying that an organization should respect and understand the needs, governance and culture of the community [2, 24, 38, 40, 165]. Further, the commercially motivated cost-saving objectives implied by the extended workforce (CO10), and related benefits, has also been observed in a number of other studies (e.g., [86, 91, 134, 157, 205, 212]). Support was also found for the survival, or more strategically, motivated objectives such as CO1 - *Build a software ecosystem* [222, 225]. Some objectives, however, were not reflected among the reviewed literature, e.g., CO3 - *Create price pressure* on third-party vendors and suppliers.

A challenge with the intangible [197], or non-monetary benefits [154] is that they may be less obvious and harder to measure and track in financial spreadsheets [154], compared to the tangible revenues and monetary benefits. In the CoMn framework, an attempt is made by presenting key benefits to each objective. Future research could strive to identify and derive suitable and actionable metrics for the different types of benefits, or objectives[9]. With such metrics in place, contribution requests and strategies can potentially become easier to motivate, execute, follow-up and learn from. Morgan and Finnegan [154] highlight

[9] See e.g., https://chaoss.community/

how organizations *" ... need to put structures and incentives in place to convert intangible asset capture into something tangible for the [organization]"*.

Objectives further align with the propositions proposed by Munir et al. [158] which states that organizations adopting OSS in their tools and infrastructure setups may benefit through reduced product development costs and time-to-market, as well as increased product and process innovation. Considering Munir et al's [158] classification of why and when organizations adopt and share software as OSS, all three case organizations can be classified as having a proactive approach with the main focus on building symbiotic relationships through their OSS engagements. Similar to Sony Mobile [158], these organizations can hence be seen as mature players where the use and sharing of OSS are motivated by business rationale.

When comparing the two private case organizations, CaseOrg1 and 2, against the public case organization CaseOrg3, many of the objectives overlapped, although the expected key benefits may differ. For example, regarding CO4 which is focused on createing or replacing an existing industry or community standard, CaseOrg 1 and 2 see value in being able to steer the direction of a community or industry, and potentially disrupting competitors by commoditizing a certain technology [212]. CaseOrg3, on the other hand, views the key benefit as improving competition in the industry or market and potentially forcing the private actors to adapt. Hence, both private and public actors may view the release of OSS and commoditization of the underlying technology as a strategic tool, but in some cases for the opposite reasons.

6.2 Contribution Complexities

The contribution complexities presented in Table 5 can be viewed from the different perspectives of a contribution strategy. In regards to if a software artifact or parts of it should be shared as OSS, a common concern in literature regards the risk of sharing differentiating or in other ways sensitive IPR ((e.g., [86, 87, 96, 212, 222, 231]). This risk was also explicitly recognized in the complexities CC4 and CC6. Selective revealing, as labeled by Henkel [86], was a recognized practice in both CaseOrg1 and 2, as well as in Sony Mobile [128]. In contrast, this was not an issue for CaseOrg3. As a public sector organization, they aim to release what they find most "differentiating" in their view, and what can provide the biggest value to their ecosystem, and in the end, the citizen. Their motivation lies in their goals that are defined by the government - *to facilitate and enable matching between job-seekers and employers on the Swedish labor market*[10]. However, they do see a balance between raising the bar for private actors in their ecosystem and not hurting their business model.

Although not explicitly found in the coding, this introduces a timing factor, i.e., that the contribution or release of the software artifact is stalled until the private actors have had time to adapt. This addresses the question of when a software

[10] https://arbetsformedlingen.se/om-oss

artifact should be shared as OSS. A related complexity is that of the commoditization process and the concerned software artifact (CC5), which is also brought up in literature [28, 82, 115, 128, 212, 231]. By being early, an organization can get a first mover advantage, gain bigger influence, and potentially steer the development, e.g., within a community through a new feature [129, 157], or within an industry through a new standard or platform [222] (see e.g., contribution objective CO1 and CO4). By being too late, an organization can risk having to adapt to competing solutions or maintain the internal solution [128, 213]. Among the case organizations, both CaseOrg1 and 2 experienced this complexity. I1 from CaseOrg1 described it as a *"...trade-off between competitive edge and burdon"*, aligning with other reported observations [231].

Lastly, the question of where, i.e., if the software artifact should be contributed to an existing community, or if a new community should be established, touches on several of the complexities, many of which are found across all three case organizations. A common concern among many of the complexities is the need for, or risk of losing, control (CC2-6). In these cases, if an organization requires a certain level of control on the development direction of the software artifact, the artifact may preferably be released in a community where the organization has a high level of influence on the RE process [129,194], or as a new community [128]. A decisive factor is the external interest for the software artifact, i.e., will it be accepted within a specific community, or if there is an interest for a new community. In the former case of an existing community, having an existing influence within the community could help to create the interest (CC12) [157]. The health of the OSS community is another complexity as a community requires active contributions from its members to stay alive. These factors may be considered in a specific community strategy, which outlines the importance of a community, and what engagement and level of influence that is needed on its RE process [129].

When weighing the options of contributing to an existing community or creating a new one, the earlier may be preferred if possible. As highlighted by CC8, creating a new community may be related to a higher cost, and comes with a number of extra challenges [109, 223]. How this is best done is beyond the scope of this paper.

6.3 The CoMn Framework and the CAP Model

Sharing software as OSS may be compared to a revealing mechanism through which an organization can gain certain benefits potentially generated from the related community. However, for an organization to gain these benefits, they must first know what the benefits are, and also what the potential costs and risks are. The pros and the cons need to be identified, analyzed and weighed against each other in order to know if the potential contribution is in line with the organization's business goals, and maybe most importantly, if it is worth it. The intent with both the CAP model and the CoMn framework is to enable organizations to answer

these questions by supporting a value-based approach [9] for creating contribution strategies and creating a foundation for how the organization can potentially quantify the return on the contributions.

Even though the intent is similar, there are some differences between the CAP model and the CoMn framework. The CAP model provides a visual tool with its two-by-two matrix and step-by-step process to classify software artifacts and identify pre-defined contribution strategies that can then be adjusted. The CoMn framework extends the two dimensions of the CAP model with the two categories of contribution objectives and contribution complexities. The contribution objectives explicate different types of benefits that may be gained as a consequence of a contribution, while the contribution complexities exemplify aspects that may complicate the contribution, or in other ways imply cost or risk for the organization.

An organization may elicit those objectives and complexities relevant to them and can develop custom contribution strategy guidelines accordingly. A contribution process can then be set up with tailored contribution request forms that can help decision-makers ask the right questions, and for the person filing the request, to better motivate why the request should be accepted. The guidelines can then help to guide decision-makers in a similar manner as for the CAP model, but as a checklist when deciding on a contribution strategy for a software artifact. Based on the checklist and similar to the CAP model, examples can be created, what Kemp [107] refers to as "Do's and Dont's", which can help to guide both decision makers and those wanting to make a contribution.

Hence, while the CAP model provides an operational tool, the intention with the CoMn framework is for an organization to use it as a tool to develop their own custom contribution guidelines. The two artifacts could, therefore, be seen as complements, providing context and input to each other when setting up a contribution process. The CoMn framework provides a foundation for creating the guidelines and template questions for creating contribution request forms. The CAP model provides concrete examples of contribution strategies and related software artifacts, but also a context for how the contribution strategy guidelines may be used in a proactive and reactive approach.

Adopting these approaches may help to streamline and decentralize decision-making in the contribution process. For example., when using the reactive approach and answering to more complex contribution requests, a cross-functional group with an executive mandate may have to decide what contribution strategy to proceed with, while smaller and less complex contribution requests may be managed on a lower organizational level, e.g., by the engineering managers. In the proactive approach, i.e., when using the contribution strategy guidelines in the product planning process, the guidelines can provide support for a software product manager when deciding if e.g., certain features should be released as OSS.

7 Threats to Validity

The CoMn framework is the result of three design cycles with a problem investigation including four case organizations. To determine the framework's external validity [191], the characteristics of these organizations need to be considered as they define the problem context [228] on which the CoMn framework is based.

The three case organizations investigated in this study both have overlapping and distinct characteristics. To some extent, they provide extremes [59] to each other while still having resembling characteristics to Sony Mobile, which was investigated in the first design cycle. Considering the way they use and leverage OSS, all four organizations use it for their internal tool and infrastructure setups. In CaseOrg2, the department developing these tools are explicitly studied. Both Sony Mobile and CaseOrg1 uses OSS to enable and add value to their hardware devices. In CaseOrg1's case, OSS is also used as a basis for certain services sold to business-oriented customers. As a public agency, CaseOrg3 differs to Sony Mobile, CaseOrg1 and CaseOrg2 in that they are not driven by commercial business incentives. Instead, they wish to help private actors focus on more value-adding activities and thereby improve job-matching capabilities for employers and job-seekers. CaseOrg2 however, does not consider themselves as having any product differentiation, compared to Sony Mobile and CaseOrg1. Following the categorization by Munir et al. [158], the organizations provide a suitable sample as they all have an awareness for why they share software as OSS and may reflect on the complexities that are present, even in the case of CaseOrg3 who does not have an explicit contribution process in place.

We acknowledge the limitations of case studies and do not claim any statistical generalization [191]. However, we do believe they provide a mean to gather deep knowledge of industry practice and rationale in the problem context. By considering the case organizations' characteristics, the reader can put the CoMn framework as well as the organizations' rationale and concerns for sharing software as OSS into context. Through analytical generalization (cf. analogical inference [228]), results from this study can then be extended to cases with similar characteristics within a similar context [191]. Both similarities and dissimilarities between the source and target cases should be thoroughly be analyzed [76]. In Section 4, we provide a general description of each of the case organizations, as well as of their specific contribution processes and examples of OSS projects that they are contributing to or have released as OSS. Quotes from interviewees have been used extensively to describe the framework to provide further contextual factors that may otherwise risk being lost in the reporting of the research if abstracted by the researcher.

A threat in regards to reliability relates to that only the first author was involved in the data gathering and analysis process of the study. To minimize the risk of miss-interpretations, member-checking [50] was performed by presenting interview summaries to key interviewees at the case organizations. During the

coding process, the inductive approach infused a constant comparison between new data and existing codes, enabled by audit-trails being kept [191]. Cross-case analysis [196] was also performed as codes identified in one case were reiterated in the transcripts from the other cases, after which a final coding scheme assembled. As can be noticed in table 3, there was a saturation in number of emerging contribution and complexities. Triangulation [191] with archival data was also performed by cross-checking coding schema and basing interview questionnaires on contribution request forms from seven organizations.

After the final coding was performed in the third design cycle, the contribution objectives and complexities were presented and discussed with I1 from CaseOrg1, I7 and I8 from CaseOrg2 and I15 from CaseOrg3. This kind of static [79] or descriptive validation [90] is a useful approach in the artifact validation phase to gain feedback and refine the artifact before it is transferred or implemented in a real-world problem context [79, 228].

I1 from CaseOrg1 confirmed identified objectives and complexities. I1 added further facets to e.g. CC6, that patents may also provide a source of license revenues in other cases which should also be considered. I1 found the framework useful and that *"it really fits into [CaseOrg1's] strategy which is, we are trying to streamline the entire [contribution] process so that we are asking the right questions and we want to get to a point where we are able to approve everything online and don't need to have a meeting. Because if these questions are answered correctly then we should be able to even have an AI/ML-based algorithm which says, the risk is 10% so it is OK to approve it"*. I1 also adds that a contribution may be *"more things than just code"*, exemplifying evangelizing, technical writing, writing bug reports, sharing of knowledge and experience, as well as test cases and design documentation. We agree with I1 that a contribution may be many things, but choose to limit the scope to the sharing of software artifacts as OSS. We view these artifacts as internally developed software functionality of different size and complexity, e.g., bug-fixes and features to frameworks, projects, and products, including e.g., related test cases and documentation. Other activities we believe should be covered by a community strategy that specifies how an organization should engage with a specific OSS community [129].

I7 and I8 from CaseOrg2 found the framework valuable and saw value in comparing with other organizations. They believed that the framework can be used as an eye-opener to those within the organization that are skeptic to OSS. Both interviewees agreed with and verified the relevance of all of the identified objectives and contributions. They also added support for CC5 and CC15 and a refined description to CO7.

I15 from CaseOrg3 confirmed the identified objectives and complexities, while also adding support for CC4 and CC5. In regards to the latter complexity, I15 highlights how it also connects to CC4 and how they strive to share software artifacts that are differentiating in order to help businesses focus on more value-adding activities

8 Conclusion

Motives for why an organization would wish to share its software as OSS can be both idealistic, strategic and commercial, and include both monetary as non-monetary benefits. These benefits may however not be obvious, and to realize them, an organization may need to consider what they share and how. A recognized risk is giving away differentiating functionality or in other ways sensitive IPR. Timing is also an important aspect as being too slow could imply that other solutions getting adopted. Target community is a third concern as there e.g., can be diverging agendas in an existing community, and creating a new community is a substantially bigger investment than contributing to an existing.

Hence, a number of complexities exist relating to the decision if and how a software artifact should be shared as OSS. Some of these may not be relevant to all organizations. Contribution processes can, for example, be more liberal in organizations or business units where the OSS used is not considered to provide a competitive advantage [157]. Contextual factors also play a role in terms of what benefits that matter for an organization and motivate why a contribution should be made [128]. By not considering relevant complexities or benefits, an organization may risk making a contribution that could be hurtful, or on the opposite, block a contribution that could have been beneficial for the organization and its business goals [231]. Organizations hence need to link their business goals with their contribution strategies, adopting what Aurum and Wohlin [9] refers to as a value-based approach.

To address this problem context, a Contribution Management (CoMn) framework is proposed. To achieve alignment between business goals and contribution strategies, software-intensive organizations may use the CoMn framework to develop contribution strategy guidelines and related contribution processes for internally developed software artifacts. A contribution strategy answers the questions if a software artifact (e.g., a feature or project) or parts of it should be released as OSS, when in time, and if it should be contributed to an existing OSS community, or if a new community should be established. The framework was created using a design science research approach with a multiple-case study of three case organizations, along with contribution request forms from seven different organizations. The framework contributes to the understanding of the problem context, as well as with design knowledge that can be used to design solutions to a problem instance. Identified objectives and complexities align with earlier work [84, 93, 128, 158, 159, 197], while also adding new perspectives to the contribution strategy decision process.

This study has focused on the design cycle of the design science research where an artifact is validated in a modeled version of the problem context [228]. Future work should look to implement and evaluate the framework in a real-world problem context, e.g., through technical action research [229]. Considering knowledge questions such as what factors may motivate an organization to share software

as OSS and related complexities, these should be further investigated, e.g., via a survey including a broader spectrum of organizations within the problem context. Specific focus should be given to metrics that may be used for quantifying the potential benefits proposed in the objectives, as well as the risks and costs implied by the complexities. Such metrics could help organizations arrive at key performance indicators such as a return on a contribution, which could potentially help to make decisions comparable, measurable, and easier to motivate. This could further help organizations in automating their contribution processes and thereby potentially lowering the threshold for developers to contribute, and in the end hopefully help the organization to more effectively reach their contribution objectives.

9 Appendix: Questionnaire - Second Design Cycle

Demographics:

- What is your job title?
- How many years of experience do you have in your current and similar roles?
- Could you, in short, describe your daily work and responsibilities?
- Could you, in short, describe your experience in working with OSS communities?

Contribution strategy:

- Do you, in any way, consider what you contribute and reveal as open source? If yes, how? Is it formalized in any way? How could this be improved/otherwise done? What connections do you see to a company's ROI and competitive edge?
- How would commoditization affect what you contribute and not? How would it affect the timing of when you contribute?
- How would a feature's profit for the company affect what is contributed and not? How would it affect the timing of when you contribute? How could this aspect be judged or quantified?
- How would it affect if the feature or the knowledge and technology behind it is hard to acquire or control? How could this aspect be judged or quantified?
- How would you reason in regard to if and when to contribute, given that a feature results in a:

  - high profit for your company, and is critical to maintain control over?
  - low profit for your company, and is critical to maintain control over?
  - high profit for your company, and not critical to maintain control over?
  - low profit for your company, and is not critical to maintain control over?

- What other aspects would affect your decisions?
- How would your decisions affect how you engage and invest in the community where the feature is to be contributed to?
- How are policies or decisions regarding what to contribute communicated today? How could this be improved/otherwise done?
- Is there anything I have missed that you would like to pick up on? Or anything else that you would like to talk about?

10 Appendix: Questionnaire - Third Design Cycle

Demographics:

- What is your job title?
- How many years of experience do you have in your current and similar roles?
- Could you, in short, describe your daily work and responsibilities?
- Could you, in short, describe your experience in working with OSS communities?

Contribution Objectives:

- What reasons do you see to share your internally developed software as OSS? What can your organization gain out of this? How can you measure it?
- What reasons do you see in relation to competition and collaboration with external parties? Who would it benefit or hurt?
- What effect can you have on a third party by releasing software as OSS? For example, competitors or suppliers of similar proprietary products?
- Do you consider if there is a potential to create a new standard? How can such potential be measured?

- Are there any type of software that should be shared for different reasons? How would these types be characterized?

Contribution Complexities:

- What aspects should you take into consideration in the decision of sharing software as open source or not?
- What consideration should be taken to how software relates to your value proposition and how it affects your revenues? For example, if the software is shipped with your product or if it is used to build or deliver your product/service?
- How is a decision affected if the software contains differentiating parts?
- How is a decision affected if the software contains patents or parts that could be patented?
- Is there any type of sensitive information within the software that could complicate a contribution?
- How should any competitive advantages be taken into consideration?
- How do timing and commoditization affect such aspects and any future decisions? How do you consider the risk of competing solutions being released before yours?
- How do you consider internal restrictions in terms of budget and resources? What costs do you see related to releasing software as OSS? How do these costs affect the decision?
- Do you see any other business related aspects, impediments or risks that should be taken into consideration?
- What consideration should be taken in regards to how the software is used and integrated into your organization and your product/services?
- What consideration should be taken to security-related aspects?
- How do you consider the software's generalizability and potential to extend?
- Do you see any other technical aspects, impediments or risks that should be taken into consideration?
- How is a decision affected in terms of the need to control the development of the software?
- How do you consider existing alternatives (OSS and proprietary)?

- What alternatives to you see if there is a shallow external interest from a community?
- What factors affect your decision when choosing to contribute to an existing community or creating a new community?
- How do you consider the health and sustainability of a community?

11 Appendix - Questions for Contribution Request Forms

Table 6: Overview of categorized questions focused on describing the contribution (Q1-6), the motive of the contribution (Q7), potential risks (Q8-13) and costs (Q14-15). Questions Q16-21 are focused on if the contribution should go to an existing community, or if a new community should be established. Each category is linked to related contribution objectives and complexities.

ID	Area	Example questions or statements	Org	COs CCs
Description of the contribution				
Q1	Goals and intention	• Provide a general description of the contribution. • What are the main objectives of the [contribution]?	O2	
Q2	Functionality	• Detailed description of the functionality	O3	
Q3	Size and complexity	• Major (an entirely new software component that contains a substantial amount of IP). • Medium (modification/extension of an already existing OSS, such as a Linux device driver, or an entirely new though modestly sized software component). • Minor (small and relatively simple modification of existing software, such as a bug fix or an error correction, not containing new functionality).	O1	
Q4	Internal dependencies	• Is the requested contribution currently used in the company? If so, where and how?	O5	CC4, CC9

Q5	Origin of source code	• Code is written from scratch specifically for the release. • Contribution includes code that was proprietary to the company (included in previous products etc). • Contribution includes third party proprietary code.	O1	
Q6	Alternatives	• Are there any similar projects from the company, or elsewhere? • Describe why such alternatives are not suitable. • If a similar project already exists, it is more powerful to team up, even if your competitor is driving it because collaboration is a critical part of OSS communities.	O1, O3, O7	CC15
Motives of the contribution				
Q7	Business rationale	• What is the business reason for making this contribution? • How will this impact revenue? • Describe any business and technical reasons supporting the release of the contribution. • A competitive desire to disrupt incumbents. • Create an opening to compete in a new field. • An internal need to reduce cost through shared software development. • Manage commoditization by raising the technology baseline. • A desire among multiple partners to work together in a neutral way. • Drive adoption with a platform and ecosystem. • Improved recruitment and on-boarding of new employees. • Improved reputation among customers and in community.	O1, O2, O4, O5, O6, O7	CO1, CO2, CO4, CO6, CO9, CO10, CC2
Risks of the contribution				

Q8	Novelty and competitive edge	• Does the IP give the company a competitive edge? • Does this project/contribution contain any new or novel ideas? If so, what? • Is the project going to be something transformative? Is it something we can showcase? • Does it include anything linked to the company's [competitive edge]?	O3, O5, O6, O7	CC4
Q9	License and obligations	• Under which OSS license should the contribution be released and why? For changes in existing OSS, it is often required that the release is made under the same OSS license. Summarize the obligations of the chosen license. • Ensure full compliance with OSS licenses.	O1, O7	CC16
Q10	Existing IPR, patents and copyright	• Will this have an impact on our IP portfolio? Intangibles? • Are there any copyright issues related to the contribution? Which copyright statement to use?	O1, O2	CC6
Q11	New inventions	• Does the contribution include any new inventions that are eligible for a patent? • Does the contribution contain any design or algorithm that would not be obvious for a skilled engineer?	O1	CC6
Q12	Security risks	• Can the software expose any sensitive information about our hardware which should not be published? • Would sharing the contribution with others outside of the company expose the company (e.g., networks, products, services, customers) to a security risk? • Does the code collect any data about users [or] transmit to us or affect or transmit third-party content?	O4, O5, O6	CC7
Q13	External impact	• Is there impact on consortia and partnerships? • Who will it hurt? Whom will it help?	O2	CO1-12, CC11
Costs of the contribution				

Q14	Implementation and Release	• Implementation cost in man weeks. • What costs are associated with releasing the contribution?	O1	CC8
Q15	Maintenance and support	• How much time or money will the company spend on maintenance and support to the community after the code is contributed? • How much time will this project require to maintain? Will the cost increase/decrease with time?	O1, O5, O6	CC8
Q16	Why not existing community	• No comparable/healthy OSS projects exist. • Unsatisfactory license or IP policy in existing projects. • Project would exceed the size, scope, or scale of existing projects. • New project needs a standalone brand identity.	O2	CC14, CC16
Q17	Community host	• Does the company prefer a specific consortium host? Should this be a standalone project?	O2	
Q18	Control and governance	• Are we comfortable with the technical implications of releasing control?	O2	CC12
Q19	External interest, awareness and reach	• What kind of participation do we reasonably expect? Who will participate? Who else needs this? Why would other companies want to join? • Do we have the people and partners we need already on-board? Can we get them to participate?	O2	CC11
Q20	Company's responsibility	• If the code will be delivered to the OSS community, will the company work actively with the community? • Will the company maintain the software, or just passively execute the contribution (without allocating maintenance resources)?	O1	CC12
Q21	Existing community health	• How long has that project been in existence? • How active is the project? • Does the company or do other large organizations use or contribute to that project, and if so for how long?	O5	CO2, CC14

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*, 32(2):385–409, 2008.

[3] Joan E. van Aken. Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules. *Journal of Management Studies*, 41(2):219–246, 2004.

[4] Thomas A. Alspaugh and Walt Scacchi. Ongoing software development without classical requirements. In *21st IEEE International Requirements Engineering Conference*, RE'13, pages 165–174, Rio de Janeiro, Brazil, July 2013. IEEE.

[5] Carina Alves, Joyce Oliveira, and Slinger Jansen. Understanding Governance Mechanisms and Health in Software Ecosystems: A Systematic Literature Review. In Slimane Hammoudi, Michał Śmiałek, Olivier Camp, and Joaquim Filipe, editors, *Enterprise Information Systems*, pages 517–542, Cham, 2018. Springer International Publishing.

[6] Morten Andersen-Gott, Gheorghita Ghinea, and Bendik Bygstad. Why do commercial companies contribute to open source software? *International Journal of Information Management*, 32(2):106–117, 2012.

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

[8] Aybüke Aurum and Claes Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954, 2003.

[9] Aybüke Aurum and Claes Wohlin. A Value-Based Approach in Requirements Engineering: Explaining Some of the Fundamental Concepts. In

Pete Sawyer, Barbara Paech, and Patrick Heymans, editors, *Requirements Engineering: Foundation for Software Quality*, pages 109–115, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[10] Alfred Baars and Slinger Jansen. A Framework for Software Ecosystem Governance. In Michael A. Cusumano, Bala Iyer, and N. Venkatraman, editors, *Software Business*, pages 168–180, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

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

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

[13] 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.

[14] 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.

[15] Willem Bekkers, Inge van de Weerd, Marco Spruit, and Sjaak Brinkkemper. A Framework for Process Improvement in Software Product Management. In Andreas Riel, Rory O'Connor, Serge Tichkiewitch, and Richard Messnarz, editors, *Systems, Software and Services Process Improvement*, pages 1–12, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[16] Tanmay Bhowmik and Anh Quoc Do. Refinement and resolution of just-in-time requirements in open source software and a closer look into non-functional requirements. *Journal of Industrial Information Integration*, 2018.

[17] 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.

[18] 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.

[19] Christian Bird and Nachiappan Nagappan. Who? Where? What?: Examining Distributed Development in Two Large Open Source Projects. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, MSR'12, pages 237–246, Piscataway, NJ, USA, 2012. IEEE Press.

[20] Elizabeth Bjarnason, Michael Unterkalmsteiner, Emelie Engström, and Markus Borg. An Industrial Case Study on Test Cases as Requirements. In Casper Lassenius, Torgeir Dingsøyr, and Maria Paasivaara, editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 27–39, Cham, 2015. Springer International Publishing.

[21] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[22] Jan Bosch. Achieving Simplicity with the Three-Layer Product Model. *Computer*, 46(11):34–39, Nov 2013.

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

[24] Simon Butler, Jonas Gamalielsson, Björn Lundell, Per Jonsson, Johan Sjöberg, Anders Mattsson, Niklas Rickö, Tomas Gustavsson, Jonas Feist, and Stefan Landemoo. An investigation of work practices used by companies making contributions to established OSS projects. In *40th International Conference on Software Engineering: Software Engineering in Practice*, ICSE'18, pages 201–210, Gothenburg, Sweden, May 2018. IEEE.

[25] 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.

[26] Eugenio Capra and Anthony I. Wasserman. A Framework for Evaluating Managerial Styles in Open Source Projects. In Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, and Giancarlo Succi, editors, *Open Source Development, Communities and Quality*, pages 1–14, Boston, MA, 2008. Springer US.

[27] John Wilmar Castro Llanos and Silvia Teresita Acuña Castillo. Differences between Traditional and Open Source Development Activities. In Oscar Dieste, Andreas Jedlitschka, and Natalia Juristo, editors, *Product-Focused Software Process Improvement*, pages 131–144, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[28] Jonathan P. Caulkins, Gustav Feichtinger, Dieter Grass, Richard F. Hartl, Peter M. Kort, and Andrea Seidl. When to make proprietary software open source. *Journal of Economic Dynamics and Control*, 37(6):1182 – 1194, 2013.

[29] InduShobha Chengalur-Smith, Saggi Nevo, and Pindaro Demertzoglou. An empirical analysis of the business value of open source infrastructure technologies. *Journal of the Association for Information Systems*, 11(11):708, 2010.

[30] Henry Chesbrough. *Open Innovation: The new imperative for creating and profiting from technology*. Harvard Business School Press, Boston, MA., 2003.

[31] Henry Chesbrough. Business model innovation: It's not just about technology anymore. *Strategy & Leadership*, 35(6):12–17, 2007.

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

[33] Henry Chesbrough, Wim Vanhaverbeke, and Joel West. *Open Innovation: Researching a new paradigm*. Oxford University Press, 2006.

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

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

[36] 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.

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

[38] 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.

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

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

[41] Daniela Damian. Stakeholders in global requirements engineering: Lessons learned from practice. *IEEE Software*, 24(2):21–27, 2007.

[42] Daniela Damian, Irwin Kwan, and Sabrina Marczak. Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people. In *Collaborative software engineering*, pages 57–76. Springer, 2010.

[43] 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*, RE'07, pages 59–68, Dehli, India, October 2007. IEEE.

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

[45] Carlos M. DaSilva and Peter Trkman. Business model: What it is and what it is not. *Long Range Planning*, 47(6):379–389, 2014.

[46] Paul B. De Laat. Governance of open source software: State of the art. *Journal of Management & Governance*, 11(2):165–177, 2007.

[47] Ivan De Noni, Andrea Ganzaroli, and Luigi Orsi. The evolution of OSS governance: A dimensional comparative analysis. *Scandinavian Journal of Management*, 29(3):247–263, 2013.

[48] Swanand J. Deodhar, KBC. Saxena, Rajen K. Gupta, and Mikko Ruohonen. Strategies for software-based hybrid business models. *The Journal of Strategic Information Systems*, 21(4):274–294, 2012.

[49] Jingshu Du, Bart Leten, Wim Vanhaverbeke, and Henry Lopez-Vega. When research meets development: antecedents and implications of transfer speed. *Journal of Product Innovation Management*, 31(6):1181–1198, 2014.

[50] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008.

[51] Evert Eckhardt, Erwin Kaats, Slinger Jansen, and Carina Alves. The Merits of a Meritocracy in Open Source Software Ecosystems. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 7:1–7:6, New York, NY, USA, 2014. ACM.

[52] 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.

[53] Ellen Enkel, Oliver Gassmann, and Henry Chesbrough. Open R&D and Open Innovation: Exploring the phenomenon. *R&D Management*, 39(4):311–316, 2009.

[54] Neil Ernst and Gail C. Murphy. Case studies in just-in-time requirements analysis. In *2nd International Workshop on Empirical Requirements Engineering*, EmpiRE'12, pages 25–32, Chicago, IL, USA, Sep. 2012. IEEE.

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

[56] Daniel Mendez Fernandez, Birgit Penzenstadler, Marco Kuhrmann, and Manfred Broy. A meta model for artefact-orientation: Fundamentals and lessons learned in requirements engineering. In Dorina C. Petriu, Nicolas Rouquette, and Øystein Haugen, editors, *Model Driven Engineering Languages and Systems*, number 6395 in Lecture Notes in Computer Science, pages 183–197. Springer Berlin Heidelberg.

[57] 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.

[58] Roy T. Fielding. Shared leadership in the Apache project. *Communications of the ACM*, 42(4):42–43, 1999.

[59] Bent Flyvbjerg. Five Misunderstandings about Case-Study Research. In *Qualitative Research Practice*, pages 390–404. SAGE, concise paperback edition, 2007.

[60] Oscar Franco-Bedoya, David Ameller, Dolors Costal, and Xavier Franch. Open source software ecosystems: A Systematic mapping. *Information and Software Technology*, 91:160 – 185, 2017.

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

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

[63] Samuel A. Fricker. Requirements Value Chains: Stakeholder Management and Requirements Engineering in Software Ecosystems. In Roel J. Wieringa and Anne Persson, editors, *Requirements Engineering: Foundation for Software Quality*, pages 60–66, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[64] Samuel A. Fricker. Software product management. In *Software for People*, pages 53–81. Springer, 2012.

[65] Jeff Frooman. Stakeholder influence strategies. *Academy of Management Review*, 24(2):191–205, 1999.

[66] Jonas Gamalielsson and Björn Lundell. Sustainability of open source software communities beyond a fork: How and why has the libreoffice project evolved? *Journal of Systems and Software*, 89:128–145, 2014.

[67] 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.

[68] Francisco José García-Peñalvo and Alicia García-Holgado. *Open Source Solutions for Knowledge Management and Technological Ecosystems*. IGI Global, 2017.

[69] Vahid Garousi, Dietmar Pfahl, João M. Fernandes, Michael Felderer, Mika Mäntylä, David Shepherd, Andrea Arcuri, Ahmet Coşkunçay, and Bedir Tekinerdogan. Characterizing industry-academia collaborations in software engineering: evidence from 101 projects. *Empirical Software Engineering*, Apr 2019.

[70] Oliver Gassmann and Ellen Enkel. Towards a Theory of Open Innovation: Three Core Process Archetypes. In *R&D Management Conference (RADMA) 2004*, July 2004.

[71] Annabelle Gawer. Bridging differing perspectives on technological platforms: Toward an integrative framework. *Research Policy*, 43(7):1239–1249, 2014.

[72] Annabelle Gawer and Michael A. Cusumano. Industry platforms and ecosystem innovation. *Journal of Product Innovation Management*, 31(3):417–433, 2014.

[73] Cees Gelderman and Arjan Weele. Handling measurement issues and strategic directions in Kraljic's purchasing portfolio model. *Journal of Purchasing and Supply Management*, 9(5 - 6):207 – 216, 2003.

[74] 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.

[75] Matt Germonprez, Julie E. Kendall, Kenneth E. Kendall, and Brett Young. Collectivism, creativity, competition, and control in open source software development: Reflections on the emergent governance of the SPDX working group. *International Journal of Information Systems and Management*, 1(1-2):125–145, 2014.

[76] Smita Ghaisas, Preethu Rose, Maya Daneva, Klaas Sikkel, and Roel J. Wieringa. Generalizing by similarity: Lessons learnt from industrial case studies. In *Proceedings of the 1st International Workshop on Conducting Empirical Studies in Industry*, CESI '14, pages 37–42, Piscataway, NJ, USA, 2013. IEEE Press.

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

[78] Jesús M. González-Barahona, Daniel Izquierdo-Cortazar, Stefano Maffulli, and Gregorio Robles. Understanding How Companies Interact with Free Software Communities. *IEEE software*, 30(5):38–45, 2013.

[79] Tony Gorschek, Per Garre, Stig Larsson, and Claes Wohlin. A model for technology transfer in practice. *IEEE software*, 23(6):88–95, 2006.

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

[81] Robert A. Hanneman and Mark Riddle. *Introduction to social network methods*. University of California Riverside, 2005.

[82] Ernan Haruvy, Suresh P Sethi, and Jing Zhou. Open source development with a commercial complementary product or service. *Production and Operations Management*, 17(1):29–43, 2008.

[83] Lile P. Hattori and Michele Lanza. On the nature of commits. In *Proceedings of the 23rd International Conference on Automated Software Engineering*, ASE'08, pages III–63–III–71, Piscataway, NJ, USA, 2008. IEEE Press.

[84] Øyvind Hauge, Claudia Ayala, and Reidar Conradi. Adoption of open source software in software-intensive organizations - a systematic literature review. *Information and Software Technology*, 52(11):1133 – 1154, 2010.

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

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

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

[88] 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.

[89] Alan R. Hevner. A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19(2):4, 2007.

[90] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS quarterly*, 28(1):75–105, March 2004.

[91] Helena Holmström Olsson and Jan Bosch. From ad hoc to strategic ecosystem management: The Three-Layer Ecosystem Strategy Model (TeLESM). *Journal of Software: Evolution and Process*, 29(7):e1876, 2017.

[92] Liaquat Hossain, Andrè Wu, and Kenneth Chung. Actor centrality correlates to project based coordination. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 363–372. ACM, 2006.

[93] Martin Höst and Alma Orucevic-Alagic. A systematic review of research on open source software in commercial software product development. *Information and Software Technology*, 53(6):616 – 624, 2011.

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

[95] 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.

[96] Netta Iivari, Henrik Hedberg, and Tanja Kirves. Usability in Company Open Source Software Context - Initial Findings from an Empirical Case Study. In Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, and Giancarlo Succi, editors, *Open Source Development, Communities and Quality*, pages 359–365, Boston, MA, 2008. Springer US.

[97] Slinger Jansen, Sjaak Brinkkemper, and Anthony Finkelstein. Business network management as a survival strategy: A tale of two software ecosystems. In *Proccedings of the 1st International Workshop on Software Ecosystems*, pages 34–48. Citeseer, 2009.

[98] 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.

[99] Slinger Jansen and Michael A. Cusumano. *Software ecosystems: analyzing and managing business networks in the software industry*, chapter Defining software ecosystems: a survey of software platforms and business network governance, pages 13–28. Edward Elgar Publishing, 2013.

[100] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. A sense of community: A research agenda for software ecosystems. In *31st International Conference on Software Engineering - Companion Volume*, ICSE'09, pages 187–190, Vancouver, BC, Canada, May 2009. IEEE.

[101] Chris Jensen and Walt Scacchi. Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study. In *29th International Conference on Software Engineering*, ICSE'07, pages 364–374, Washington, DC, USA, May 2007. IEEE.

[102] Chris Jensen and Walt Scacchi. Governance in open source software development projects: A comparative multi-level analysis. In Pär Ågerfalk, Cornelia Boldyreff, Jesús M. González-Barahona, Gregory R. Madey, and John Noll, editors, *Open Source Software: New Horizons*, pages 130–142, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[103] Mitchell Joblin, Sven Apel, Claus Hunsen, and Wolfgang Mauerer. Classifying developers into core and peripheral: An empirical study on count and network metrics. In *39th International Conference on Software Engineering*, ICSE'17, pages 164–174, Buenos Aires, Argentina, May 2017. IEEE.

[104] Gerry Johnson, Kevan Scholes, and Richard Whittington. *Exploring corporate strategy: Text & cases*. Pearson Education, 2008.

[105] Jaap Kabbedijk and Slinger Jansen. Steering insight: An exploration of the ruby software ecosystem. In Björn Regnell, Inge van de Weerd, and Olga De Troyer, editors, *Software Business*, pages 44–55, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[106] 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.

[107] Richard Kemp. Open Source Software (OSS) governance in the organisation. *Computer Law & Security Review*, 26(3):309–316, 2010.

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

[109] Terhi Kilamo, Imed Hammouda, Tommi Mikkonen, and Timo Aaltonen. From proprietary to open source - Growing an open source ecosystem. *Journal of Systems and Software*, 85(7):1467–1478, 2012.

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

[111] Hans-Bernd Kittlaus and Samuel A. Fricker. *Software Product Management: The ISPMA-Compliant Study Guide and Handbook*. Springer, 2017.

[112] Eric Knauss, Daniela Damian, Alessia Knauss, and Arber Borici. Openness and requirements: Opportunities and tradeoffs in software ecosystems. In *22nd International Requirements Engineering Conference*, RE'14, pages 213–222, Karlskrona, Sweden, Aug 2014. IEEE.

[113] John Koenig. Seven open source business strategies for competitive advantage. *IT Manager's Journal*, 14, 2004.

[114] 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.

[115] Peter M. Kort and Georges Zaccour. When should a firm open its source code: a strategic analysis. *Production and Operations Management*, 20(6):877–888, 2011.

[116] Peter Kraljic. Purchasing must become supply management. *Harvard Business Review*, 61(5):109–117, 1983.

[117] Jaison Kuriakose and Jeffrey Parsons. How do Open Source Software (OSS) developers practice and perceive requirements engineering? An empirical study. In *5th International Workshop on Empirical Requirements Engineering*, EmpiRE'15, pages 49–56, Ottawa, ON, Canada, Aug 2015. IEEE.

[118] 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.

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

[120] Paula Laurent and Jane Cleland-Huang. Lessons Learned from Open Source Projects for Facilitating Online Requirements Processes. In Martin Glinz and Patrick Heymans, editors, *Requirements Engineering: Foundation for Software Quality*, pages 240–255, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[121] 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.

[122] 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.

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

[124] Josh Lerner and Jean Tirole. Some simple economics of open source. *The Journal of Industrial Economics*, 50(2):197–234, 2002.

[125] 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.

[126] Johan Linåker, Maria Krantz, and Martin Höst. On Infrastructure for Facilitation of Inner Source in Small Development Teams. In Andreas Jedlitschka, Pasi Kuvaja, Marco Kuhrmann, Tomi Männistö, Jürgen Münch, and Mikko Raatikainen, editors, *Product-Focused Software Process Improvement*, pages 149–163, Cham, 2014. Springer International Publishing.

[127] Johan Linåker, Hussan Munir, Per Runeson, Björn Regnell, and Claes Schrewelius. A survey on the perception of innovation in a large product-focused software organization. In João M. Fernandes, Ricardo J. Machado, and Krzysztof Wnuk, editors, *Software Business*, pages 66–80, Cham, 2015. Springer International Publishing.

[128] Johan Linåker, Hussan Munir, Krzysztof Wnuk, and Carl-Eric Mols. Motivating the contributions: An open innovation perspective on what to share as open source software. *Journal of Systems and Software*, 135:17–36, 2018.

[129] Johan Linåker, Björn Regnell, and Daniela Damian. A community strategy framework - how to obtain influence on requirements in meritocratic open source software communities? *Information and Software Technology*, 2019.

[130] Johan Linåker, Björn Regnell, and Daniela Damian. A method for analyzing stakeholders' influence on an open source software ecosystem's requirements engineering process. *Requirements Engineering*, Apr 2019.

[131] 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*, ICSSP'15, pages 208–212. ACM, Aug. 2015.

[132] Johan Linåker, Patrick Rempel, Björn Regnell, and Patrick Mäder. How firms adapt and interact in open source ecosystems: Analyzing stakeholder influence and collaboration patterns. In Maya Daneva and Oscar Pastor, editors, *Requirements Engineering: Foundation for Software Quality*, pages 63–81, Cham, 2016. Springer International Publishing.

[133] Juho Lindman and Imed Hammouda. Investigating Relationships Between FLOSS Foundations and FLOSS Projects. In Federico Balaguer, Roberto Di Cosmo, Alejandra Garrido, Fabio Kon, Gregorio Robles, and Stefano Zacchiroli, editors, *Open Source Systems: Towards Robust Practices*, pages 14–22, Cham, 2017. Springer International Publishing.

[134] Juho Lindman, Juha-Pekka Juutilainen, and Matti Rossi. Beyond the business model: Incentives for organizations to publish software source code. In Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman, editors, *Open Source Ecosystems: Diverse Communities Interacting*, pages 47–56, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[135] 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.

[136] Björn Lundell, Brian Lings, and Edvin Lindqvist. Open source in Swedish companies: where are we? *Information Systems Journal*, 20(6):519–535, 2010.

[137] Björn Lundell, Brian Lings, and Anna Syberfeldt. Practitioner perceptions of Open Source software in the embedded systems area. *Journal of Systems and Software*, 84(9):1540–1549, 2011.

[138] Hanna Mäenpää, Simo Mäkinen, Terhi Kilamo, Tommi Mikkonen, Tomi Männistö, and Paavo Ritala. Organizing for openness: six models for developer involvement in hybrid OSS projects. *Journal of Internet Services and Applications*, 9(1):17, 2018.

[139] 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.

[140] Joan Magretta. Why business models matter. *Harvard Business Review*, 80(5):86–92, 2002.

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

[142] Sabrina Marczak, Daniela Damian, Ulrike Stege, and Adrian Schröter. Information Brokers in Requirement-Dependency Social Networks. In *16th International Requirements Engineering Conference*, RE'08, pages 53–62, Catalunya, Spain, Sept 2008. IEEE.

[143] M. Lynne Markus. The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management & Governance*, 11(2):151–163, 2007.

[144] A. Mendelow. Stakeholder mapping. In *Proceedings of the 2nd International Conference on Information Systems*. MA Cambridge, 1991.

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

[146] Alastair Milne and Neil Maiden. Power and politics in requirements engineering: embracing the dark side? *Requirements Engineering*, 17(2):83–98, 2012.

[147] 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.

[148] 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.

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

[150] Jefferson Seide Molléri, Kai Petersen, and Emilia Mendes. Survey guidelines in software engineering: An annotated review. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM'16, pages 58:1–58:6, New York, NY, USA, 2016. ACM.

[151] Carl-Eric Mols, Krzysztof Wnuk, and Johan Linåker. The open source officer role – experiences. In Federico Balaguer, Roberto Di Cosmo, Alejandra Garrido, Fabio Kon, Gregorio Robles, and Stefano Zacchiroli, editors, *Open Source Systems: Towards Robust Practices*, pages 55–59, Cham, 2017. Springer International Publishing.

[152] Lorraine Morgan, Joseph Feller, and Patrick Finnegan. Exploring Inner Source as a Form of Intra-Organisational Open Innovation. In *ECIS 2011 Proceedings*, number 151, pages 1–12. AISeL, 2011.

[153] 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.

[154] Lorraine Morgan and Patrick Finnegan. Beyond free software: An exploration of the business value of strategic open source. *The Journal of Strategic Information Systems*, 23(3):226–238, 2014.

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

[156] 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. ACM.

[157] Hussan Munir, Johan Linåker, Krzysztof Wnuk, Per Runeson, and Björn Regnell. Open innovation using open source tools: a case study at sony mobile. *Empirical Software Engineering*, 23(1):186–223, Feb 2018.

[158] Hussan Munir, Per Runeson, and Krzysztof Wnuk. A theory of openness for software engineering tools in software organizations. *Information and Software Technology*, 97:26 – 45, 2018.

[159] Hussan Munir, Krzysztof Wnuk, and Per Runeson. Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering*, 21(2):684–723, 2016.

[160] Frank Nagle. Learning by contributing: gaining competitive advantage through contribution to crowdsourced public goods. *Organization Science*, 29(4):569–587, 2018.

[161] 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*, IWPSE'02, pages 76–85, Orlando, Florida, May 2002. ACM.

[162] Robert Newcombe. From client to project stakeholders: a stakeholder mapping approach. *Construction Management and Economics*, 21(8):841–848, 2003.

[163] Mark Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1):016132, 2001.

[164] Mark Newman. *Networks: An introduction*. Oxford University Press, 2010.

[165] Anh Nguyen Duc, Daniela S. Cruzes, Geir K. Hanssen, Terje Snarby, and Pekka Abrahamsson. Coopetition of Software Firms in Open Source Software Ecosystems. In Arto Ojala, Helena Holmström Olsson, and Karl Werder, editors, *Software Business*, pages 146–160, Cham, 2017. Springer International Publishing.

[166] Anh Nguyen-Duc, Daniela S. Cruzes, Snarby Terje, and Pekka Abrahamsson. Do Software Firms Collaborate or Compete? A Model of Coopetition in Community-initiated OSS Projects. *e-Informatica Software Engineering Journal*, 13(1):37–62, 2019.

[167] John Noll. Innovation in Open Source Software Development: A Tale of Two Features. In Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti, editors, *Open Source Development, Adoption and Innovation*, pages 109–120, Boston, MA, 2007. Springer US.

[168] Linus Nyman and Juho Lindman. Code forking, governance, and sustainability in open source software. *Technology Innovation Management Review*, 3(1):7–12, 2013.

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

[170] Michael Olsen. *Open sources 2.0: The continuing evolution*, chapter Dual Licensing. O'Reilly Media, Inc., 2005.

[171] Siobhán O'Mahony. Nonprofit Foundations and Their Role in Community-Firm Software Collaboration. In *Perspectives on Free and Open Source Software*, pages 393–414. MIT Press, Cambridge, MA, 2005.

[172] Siobhán O'Mahony. The governance of open source initiatives: what does it mean to be community managed? *Journal of Management & Governance*, 11(2):139–150, 2007.

[173] Siobhán O'Mahony and Beth A. Bechky. Boundary organizations: Enabling collaboration among unexpected allies. *Administrative Science Quarterly*, 53(3):422–459, 2008.

[174] Siobhán O'Mahony and Fabrizio Ferraro. The emergence of governance in an open source community. *Academy of Management Journal*, 50(5):1079–1106, 2007.

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

[176] 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.

[177] 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.

[178] 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.

[179] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007.

[180] 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.

[181] Miikka Poikselkä, Harri Holma, Jukka Hongisto, Juha Kallio, and Antti Toskala. *Voice over LTE (VoLTE)*. John Wiley & Sons, 2012.

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

[183] Dirk Riehle. The Economic Motivation of Open Source Software: Stakeholder Perspectives. *Computer*, 40(4):25–32, April 2007.

[184] Dirk Riehle. The Commercial Open Source Business Model. In Matthew L. Nelson, Michael J. Shaw, and Troy J. Strader, editors, *Value Creation in E-Business Management*, pages 18–30, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[185] Dirk Riehle. Controlling and steering open source projects. *Computer*, 44(7):93–96, 2011.

[186] Dirk Riehle. The single-vendor commercial open course business model. *Information Systems and e-Business Management*, 10(1):5–17, 2012.

[187] Colin Robson. *Real world research*, volume 3. Wiley Chichester, 2011.

[188] 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.

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

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

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

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

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

[194] Mario Schaarschmidt, Gianfranco Walsh, and Harald FO. von Kortzfleisch. How do firms influence open source software communities? A framework and empirical analysis of different governance modes. *Information and Organization*, 25(2):99–114, 2015.

[195] Stijn Schuermans, Andreas Constantinou, and Michael Vakulenko. *Assymetric Business Models: The secret weapon of software-driven companies*. Vision Mobile, 2014.

[196] Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, 1999.

[197] Shahrokh Shahrivar, Shaban Elahi, Alireza Hassanzadeh, and Gholamali Montazer. A business model for commercial open source software: A systematic literature review. *Information and Software Technology*, 103:202 – 214, 2018.

[198] Maha Shaikh and Ola Henfridsson. Governing open source software through coordination processes. *Information and Organization*, 27(2):116–135, 2017.

[199] 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.

[200] Zeena Spijkerman and Slinger Jansen. The open source software business model blueprint: A comparative analysis of 10 open source companies. In *Proceedings of the International Workshop on Software-intensive Business: Start-ups, Ecosystems and Platforms*, volume 2018, pages 128–145, 2018.

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

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

[203] 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.

[204] Margaret-Anne Storey, Emelie Engström, Martin Höst, Per Runeson, and Elizabeth Bjarnason. Using a Visual Abstract As a Lens for Communicating and Promoting Design Science Research in Software Engineering. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM'17, pages 181–186, Piscataway, NJ, USA, 2017. IEEE Press.

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

[206] Mahbubul Syeed, Juho Lindman, and Imed Hammouda. Measuring Perceived Trust in Open Source Software Communities. In Federico Balaguer, Roberto Di Cosmo, Alejandra Garrido, Fabio Kon, Gregorio Robles, and Stefano Zacchiroli, editors, *Open Source Systems: Towards Robust Practices*, pages 49–54, Cham, 2017. Springer International Publishing.

[207] David J. Teece. Business models, business strategy and innovation. *Long Range Planning*, 43(2):172–194, 2010.

[208] 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.

[209] 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.

[210] 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.

[211] George Valença and Carina Alves. A theory of power in emerging software ecosystems formed by small-to-medium enterprises. *Journal of Systems and Software*, 134:76–104, 2017.

[212] 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.

[213] 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.

[214] Dindin Wahyudin, Khabib Mustofa, Alexander Schatten, Stefan Biffl, and A. Min Tjoa. Monitoring the "health" status of open source web-engineering projects. *International Journal of Web Information Systems*, 3(1/2):116–139, 2007.

[215] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge University Press, 1994.

[216] Richard T. Watson, Marie-Claude Boudreau, Paul T. York, Martina E. Greiner, and Donald Wynn Jr. The business of open source. *Communications of the ACM*, 51(4):41–46, 2008.

[217] Florian Weikert and Dirk Riehle. A model of commercial open source software product features. In Georg Herzwurm and Tiziana Margaria, editors, *Software Business. From Physical Products to Software Services and Solutions*, pages 90–101, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

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

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

[220] Joel West. Value capture and value networks in open source vendor strategies. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, HICSS'07, pages 176–176, Waikoloa, HI, USA, Jan 2007. IEEE.

[221] Joel West and Marcel Bogers. Leveraging External Sources of Innovation: A Review of Research on Open Innovation. *Journal of Product Innovation Management*, 31(4):814–831, 2014.

[222] 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.

[223] Joel West and Siobhán O'Mahony. Contrasting Community Building in Sponsored and Community Founded Open Source Projects. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, HICSS'05, pages 196–196, Big Island, HI, USA, Jan 2005. IEEE.

[224] Joel West and Siobhán O'Mahony. The role of participation architecture in growing sponsored open source communities. *Industry and Innovation*, 15(2):145–168, 2008.

[225] Joel West and David Wood. Creating and Evolving an Open Innovation Ecosystem: Lessons from Symbian Ltd. *Available at SSRN 1532926*, 2008.

[226] 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.

[227] Roel J. Wieringa. Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology*, page 8. ACM, 2009.

[228] Roel J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[229] Roel J. Wieringa and Ayşe Moralı. Technical Action Research as a Validation Method in Information Systems Design Science. In Ken Peffers, Marcus Rothenberger, and Bill Kuechler, editors, *Design Science Research in Information Systems. Advances in Theory and Practice*, pages 220–238, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[230] Bernd W. Wirtz, Adriano Pistoia, Sebastian Ullrich, and Vincent Göttel. Business models: Origin, development and future research perspectives. *Long Range Planning*, 49(1):36–54, 2016.

[231] 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*, ESEM'12, pages 271–280, New York, NY, USA, 2012. ACM.

[232] Krzysztof Wnuk, Björn Regnell, and Brian Berenbach. Scaling Up Requirements Engineering – Exploring the Challenges of Increasing Size and Complexity in Market-Driven Software Development. In Daniel Berry and Xavier Franch, editors, *Requirements Engineering: Foundation for Software Quality*, pages 54–59, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[233] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[234] Nicole Ziegler, Oliver Gassmann, and Sascha Friesike. Why do firms give away their patents for free? *World Patent Information*, 37:19 – 25, 2014.