UNIVERSITY OF GOTHENBURG

A study on Opening Software Platforms

Bachelor of Science Thesis in the program Software Engineering and Management

MARKUS ERLACH
Sara Johansson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2016

A study on opening up Software Platforms

MARKUS. ERLACH,
SARA. JOHANSSON,

Examiner: IMED. HAMMOUDA

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2016

Abstract

This study investigates factors for opening up a software platform within an industrial setting. Using questionnaires and interviews, data was gathered and grounded theory was applied. The results showed that guidelines for how to open up a software platform are needed and that companies lacked domain knowledge.

Acknowledgment: We would like to thank our supervisor Thorsten Berger (Chalmers — University of Gothenburg, Sweden) for his guidance. We would also like to thank our partners Christoph Elsner (Siemens AG, Germany), Benedikt Schultis (Siemens AG, Germany), and Christoph Seidl (Technical University of Braunschweig, Germany) for their support in the making of our bachelor thesis.

1 Introduction

There is a significant increase of companies that have adopted a software ecosystem. J. Bosch discusses one of the most common transitions from a software product line to a software ecosystem. One of his reasons for this transition is a "a successful platform and intra-organizational software product line".[11] S. Jansen, A. Finkelstein, and S. Brinkkemper define a software ecosystem as "a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts." This definition tells us that the relationships in a software ecosystem are supported and held together by a platform.

In this thesis We follow M. Gawer and A. Cusumano (2002), and their definition of a software platform. They define it as "A foundation technology or set of components used beyond a single firm and that brings multiple parties together for a common purpose or to solve a common problem". A software platform allows for the creation of multiple different applications to be built upon it and allows for variation based off of the underlying structure of the software platform [3][8]. This can make companies become platform leaders and a platform leader will be able to influence complementary products to be created within their industry [3]. Being in a leadership position if other companies do not aid in innovating can on the other hand be quite costly. For example, Intel produces microprocessors for PC's and when developing their 64-bit processor spent a lot of time and money. Had the rest of the market not begun innovating their products, Intel would have lost an immense amount of money and time spent on development [2].

This study focuses on the platform of a software ecosystem. We aim at getting a grasp on what industries are required to do in order to go from a closed software platform to an open software platform. Since a platform is an underlying technological factor of an ecosystem we also investigate what technological aspects a software platform would need change to open up a software platform.

A closed platform is proprietary, controlled and owned by a single organization or party. The platform development is limited to internal contributions within the organization. This could be done through licensing, legal actions such as patents, copyrights or plain secrecy [13]. So what does open mean? Eisenmann et al., (2008) define a platform as open if the development, the contributions, the use of the platform, and the commercialization is not restricted. Or if all restrictions placed are reasonable and applied equally to all. Two ways of opening a platform can be seen, horizontal and vertical [20]. A horizontal approach means sacrificing some control by licensing the platform to competitors, or integrating further platform sponsors. A vertical approach means granting external developers access to the market of complementary applications [8] [9]. This is the definition of open we will use in our thesis.

As software companies with closed software platforms grow, they can choose to open up their software platform allowing external actors a chance to contribute to the development [8][9]. It is worth mentioning that a company controls the growth of its ecosystem and

it also decides exactly what external developers can access[7]. Despite this companies do not always know how to open up their software platform. They do not know which parts of their platform should be opened, how much should be opened, which technology should be changed and they do not know what factors can influence the success when opening up their company's software platform. This study was designed and carried out with the aim of answering these questions in an industrial setting.

- RQ1: How do companies successfully open up closed software platforms?
  - SQ1: What factors contribute to the opening of a software platform?
  - SQ2: How does opening up a software platform affect a company?
- RQ2: What technological changes are necessary?

2 Related Research

2.1 Software Platforms

Kilamo et al. (2011) study the problems that come with making communities for open source software that previously was closed. They looked at industrial case studies, developed tools and methods, and worked together with companies in an attempt to identify best practices for opening software in an industrial setting. Working with these companies they tried to applied the OSCOMM framework, which is a framework that helps with opening up software that was previously proprietary in three phases. Looking at the data gathered and the tools and frameworks used in this study, we can extract information that could be beneficial to us. The OSCOMM framework was used in a case study where a software company applied the OSCOMM framework with the goals of growing and expanding to reach international partners. After evaluating their use of OSCOMM, the company stated that they should have focused more on re-factoring the software for community development and that they in the future plan on investing more in marketing. Attempting to increase the size of the community so that it can reach a level of self-sustainability.

Platform-mediated networks can have several different types of roles that can all be opened to encourage participation from external sources. Looking at these factors can help us see which type of factors more greatly benefits the opening of a software platform as well as in what context they are used. Eisenmann et al. (2008) look at the different factors that either motivate decisions on opening or closing the platform. They concluded that openness occurs at multiple levels depending on whether the participation is unrestricted. Several types of strategies were formed that either grant or restrict access to the different participants. When proprietary platforms mature, they are usually opened to include new providers that contribute to the platform. As the platform grows, these new providers will usually have opinions about the way the platform is directed and try to force the platform to open its governance. A shared platform on the other hand tends to go the other way and look for a central authority to be able to settle differences and set directions. This closes a previously open platform's governance. What this means for platform governance is that forces tend to push both different types towards a hybrid style governance where control is central and handle platform technology, and the responsibility of serving users is shared. Eisenmann et al. (2008) used four different roles to show the openness of a platform. These roles are the demand-side user, the supply-side user, the platform provider and the platform sponsor. Eisenmann et al. (2008) give examples of platforms that are all successful but open at different role levels. When evaluating our survey responses, we will know that where a company decides to allow access will not give a definitive answer of where to open up a software platform. Instead it might allow us to see patterns related to which roles are open and what other factors contributed to a successful opening.

M. Anvaari and S. Jansen (2010) study how different architecture and openness of mobile platforms can affect the growth of a software ecosystem. The success of a mobile platform being open or closed can be ar-

gued depending on what aspects you see as success. Examining how open a platform is would let you understand openness strategy of a platform, which entails how much of a platform the supplier will give a user access to. Anvaari and Jansen (2010), developed a tool that looks at three different layers; applications layer, middleware layer and kernel layer. The model also shows three ways of how accessible the platform is from each layer. Using this tool they tried to identify architectural factors that would show the openness strategy behind the different mobile platforms. The model however does not take into consideration licensing aspects of the platforms. Anvaari and Jansen (2010), show that by taking licensing into consideration, you can more clearly see which parts are open and identify factors of the openness strategy. Anvaari and Jansen (2010) conducted interviews with application developers, in order to confirm the results they had gotten regarding how open the different mobile platforms are. This confirmed some of their previous results, but they also learned that application developers believe that the degree to which a mobile platform is open might be lesser than what the literature claims.

2.2 Ecosystems

It is vital that the concept of ecosystems is grasped before making any change to a software platform. Berger et al. (2014) compare the technical mechanisms of open and closed platforms. Closed platforms have mechanisms that hinder development that is not profitable. Open platforms address other aspects which allows for a high security but an easy access. Using simple dependency languages allows ecosystems that provide end users with free market assets to use mechanisms. These mechanisms shapes the open-platform design by affecting direct and indirect dependencies. Berger et al. (2014) created a conceptual framework in order to evaluate ecosystems. They found that every ecosystems type of framework assisted in directing the design of the platform. Giving clear insight on relationships, variability mechanisms, and characteristics of the ecosystem. In order to draw conclusions about the different platforms/products that the survey and interview participants have described, the connections within the ecosystem needs to be understood.

During research it is just as vitally important to investigate the failures as the successes. An opening of a software platform can fail for multiple reasons, not uncommonly it is due to failures and lacks within other aspects of the company or ecosystem. Jansen (2014) discusses an operationalization of the health of an open source ecosystem. Allowing an opportunity to investigate healthy ecosystems without starting from scratch. R. Costanza and M. Mageau (1999) define "the concept of ecosystem health as a comprehensive, multiscale, dynamic, hierarchical measure of system resilience, organization, and vigor". Part of a healthy ecosystem is choosing the appropriate development predictions of a platform. For an open platform to be healthy, it is necessary to constantly maintain and extend it to keep up with competing platforms. Jansen (2014) provides a view of healthy ecosystems so that strategical changes to an open platform can be made. Once a platform is open, it needs to stay open and evolve.

Henk, Slinger and Sjaak(2011) recognized a gap in the knowledge of using in-the-field software operations that organizations could use to improve the ecosystems quality. The thought behind this paper is to receive the same type of benefits one could achieve from opening up a closed platform. By filling in this gap those that are part of the ecosystem can make the software better and stronger, increasing the stability and health of the ecosystem. Henk, Slinger and Sjaak(2011) discusses the importance of knowing how a software actually operates and what it provides to its users. By analyzing organizations knowledge of software operations they concluded that infrastructure, utilization and propagation are the main points to improve in order for this gap to be filled. Looking at the domain knowledge of companies in our survey will allow us to better analyze the importance of this aspect when opening up a software platform. Henk, Slinger and Sjaak(2011) realized that there is a parallel connection to software operation knowledge, and to trustworthy relationships between the software ecosystem participants.

3 Methodology

3.1 Research methodology

Grounded theory is a way of creating theories that are based on analyzing data. The source of data can be anything from a document to a video file or a researcher's own experiences. The basic idea or method would be to start with a case or experience and go through it in order to develop abstract categories. So that one could create, explain and understand the data, identifying interrelationships within it. The data that is collected in this study is analyzed using grounded theory. Grounded theory offers several different types of ways to analyze data. Kathy Armas(1996) discusses line-by-line coding and focused coding which expects questions and categories to be predetermined. While Naresh R. Pandit(1996) discusses a more combined and detailed data analysis. These types are open coding, axial coding, and selective coding. Open Coding is an analytic type of coding which involves asking questions in order to gather data into different categories and sub categories. These categories can be built up again using axial coding to connect the dots between the main categories and their children, allowing for both new answers and questions to be formed. Similarly, selective coding also draws new conclusions but by connecting the main categories.

3.2 Eliciting participants

To find participants for the survey questionnaire, we needed to locate Software Engineering industry practitioners. This was done by scouring software engineering conferences and in particular conferences that had Software Engineering in Practice (SEIP) workshops. We looked through conferences that were about software engineering, software platforms and software ecosystems. Conferences that were scoured include: International Conference on Software Engineering (ICSE), International Workshop on Software Ecosystems (IWSECO) and Working IEEE/IFIP Conference on Software Architecture (WICSA). These conferences are held annually and we went back as far as 2006 looking for industry practitioners. We did not look at any particular domains like medical or automotive industries but rather industries as a whole. Limiting our data collection to any one particular industry would have limited our results and could have made our participant pool too small.

Inclusion criteria: The company needed to have opened, be in the process of opening or tried to open their software platform previously. The companies also needed to have a contact person with an email for us to be able to contact them personally. To find the most suitable participants, personnel involved with the software industry were prioritized. Participants were largely found by looking at industry committees or industry workshops at these conferences.

Exclusion criteria: Companies with contact emails that would most likely be sent to an info email were excluded since the chance of a response was deemed to be low. Companies with too little information on their software platform were also excluded because one could not ascertain whether they were opening, had opened or were trying to open their software platform. People involved in research at companies or at software engineering institutes had already largely been found and were therefore excluded.

3.3 Surveys

The questionnaires content was carefully created by deducing what type of information was needed for the research and tested by a third-party, allowing for critique and some reconstruction. The questionnaire consists of 31 questions and it is optional for the participant to add his/her name and email at the end, thus making anonymity a choice and soliciting whether the participant wanted the final report. Some questions allow the participant to answer multiple choices for the same question. The questionnaire is attached as an appendix, and the following list shows how we map the survey questions with the research questions.

- Questions 1 - 4 General properties.
- Questions 5 - 8 Technological aspects (RQ2).
- Questions 9 - 16 Reasons, process and strategy (SQ1).
- Questions 17 & 18 Technological aspects (RQ2).
- Questions 19 - 25 Affect, success and challenges (RQ1, SQ2).
- Questions 26 - 29 Technological aspects (RQ1, RQ2).
- Questions 30 - 31 Participant individual questions.
- Questions 32 - 35 Contact and final remarks.

The second part of the survey was the interview which was aimed at participants whose software platform had been opened and is still open. Based on the interviewees individual questionnaire answers, an interview guideline was created in such a way that the interview was semi-structured. Using the semi-structured method allowed the interviewer, one of our partners, to ask questions during the interview based on information gathered then and there. To arrange the meeting a formal email directed towards the participant was sent, and to encourage participation the interview was set to be 30 to 45 min long.

4 Results

Looking at the open coding data it is clear that certain patterns can be acknowledged. The data contains information regarding several different types of software platforms within industries, those that have gone from closed to opened platforms, those that designed their platform to be open and those that made certain areas within their platform more open.

After evaluating the open coding categories, certain categories were deemed unnecessary for the research and disregarded in the next step of analysis. In order to further evaluate the data and create new questions that needed answering, axial coding was used to find points of interest and interrelationships between categories and sub-categories. These new questions and points of interest created the foundation for the interview guideline draft. Our questionnaire was answered in full by 15 individual participants from different projects. Out of these 15 responses, 4 projects are open software platforms and 11 are still in the process of opening up. The participant's roles in software platform projects have been roles such as Marketing expert to Developer or Software Architect and some participants also answered that they have had multiple roles.

Role	Count	Percent %
Developer	8	50
Modeler	1	6,3
Software architect	11	68.8
Team leader	6	37.5
Project manager	5	31.3
Domain expert	3	18.8
Researcher	1	6.3
Product manager	2	6.3
Marketing expert	1	6.3
Other: Requirements engineer	1	6.3
Other: User	1	6.3
Other: System Architect	1	6.3
Other: Consultant	1	6.3

Table 1: Roles.

The experience that the participants have, ranged from 3 to 5 years up to greater than 10 years. Only 1 (6.7%) participant had 3 to 5 years of experience , 2 (13.3%) participants had between 5 and 10 years of experience and 12 (80.0%) participants had more than 10 years of experience within the software engineering field.

4.1 Results of questionnaire analysis

Industries are driven to implement an open software platform and the majority of industries that participated in the study (40%) had chosen to do a complete re-implementation of a closed platform. While 30%

chose to re-engineer, re-factor, or extend their closed platform. 10% of the participants decided to design a completely new software platform making it open from the start.

4.2 Reasons for opening up a software platform

From all the participants a total of 12 business intentions could be recognized. Innovation through extensions by third parties received a 19.5% making it the most common reason. Industries are commonly profit focused, so it is not a surprise that 12.2% participants had the business intention to increase users and 12.2% had shared cost of innovation.There were several other business intentions such as; increase attractiveness for new users, new revenue streams, new application areas, establish a unique selling point, establish a value chain, increase value for existing users, stabilize market position, increase domain knowledge in platform organization, and reduce commodity burden. Another strong factor to pushing the opening up of a software platform are technical intentions. Enable external realization of specialized requirements and realize functionality that is beyond the organization's capacity are the technical intentions of 70% of the participants. Some specific problems have arisen in several of the projects that answered the questionnaire, leading them to try and open up their software platform. The most occurring problems are; an overflow of requirements from users (33.3%), struggling with market competition (27.8%), and lack of compatibility with other platform (11.1%).

The participants benefits differed from one another. Some of the benefits include an increased domain knowledge, increased mind share in a technical domain, increased extensions and extensions implementation frequency. Other benefits include having new business opportunities, a common application framework, extended platform functionality, established integration', and for development teams the ability to develop and deliver on their own schedule. The drawbacks also differed from one another. The drawbacks included; high effort to keep interfaces stable, hard to maintain backward compatibility, negotiation between stakeholders, customers losing interest, code ownership, making decisions, varying quality of extensions, balancing internal and external business models, protecting company from architectural drivers, process-immaturity, more testing needed, and extended integration was needed.

Out of the 15 participants that completed the questionnaire, 1 respondent chose not to answer what challenges they had faced in the process of opening up their software platform. The challenges faced most often are having to restructure the architecture, introducing new technologies and being able to maintain backwards compatibility. 71.4% of all respondents stated that maintaining backwards compatibility was a challenge.

Challenge	Count	Percent %
Maintaining backwards compatibility	10	71.4
Restructuring teams	4	28.6
Restructuring architecture	9	64.3
Introducing new technologies	9	64.3
User acceptance	3	21.4
Modeling the ecosystem	1	7.1
Other:	4	28.6

Table 2: Challenges faced.

Out of the 15 participants that completed the questionnaire, 1 respondent chose not to answer what aspects they found important for sustaining the success of their open software platform. The participants personal opinions are commonly shared when it comes to sustaining the success of an open software platform. All 14 (100%) participants either agreed or strongly agreed that the software quality of the platform as well as having stable extension mechanisms are the most important aspects in sustaining the success of the open platform. 13 (92.9%) participants either agreed or strongly agreed that the software quality of the extensions are important, while 1 (7.1%) was neutral. 10 participants either agreed or strongly agreed that the quality assurance of extensions are

important while 3 (21.4%) were neutral and 1 (7.1%) disagreed. That having a large number of extensions is important was the opinion of 3 (21.4%) participants while 5 (35.7%) remained neutral and 6 (42.9%) either disagreed or strongly disagreed. 2 (14.3%) participants either agreed or strongly agreed that having a market place for extensions was important for sustaining the success. 5 (35.7%) remained neutral and 7 (50.0%) disagreed or strongly disagreed. Community management was agreed or strongly agreed upon to be important by 8 (57.1%) participants, with 4 (28.6%) being neutral and 2 (14.3%) disagreeing and strongly disagreeing.

Out of the 15 respondents who completed the questionnaire, 1 did not answer on whether he/she believed that they needed to significantly change the business model. The percentages for business model is out of 14 answers whereas the rest are calculated with a total of 15 answers. Roughly 60% agreed or strongly agreed that their business model, their architecture, their process of platform development and their organization of development needed to be changed significantly when opening their platform. The results gathered are summarized in table 3.

Verifying the quality of external extensions is shown to be done using a few different ways but is most likely aided by the documentation and support for development of extensions. Out of the 15 participants that completed the questionnaire, 5 (33.3%) stated that they use manual technical reviews, 1 use certification of contributors, 2 (13.3%) use certification of the extensions, 2 (13.3%) use certification of the development process, 3 (33.3%) utilize contracts to oblige the contributors to use certain quality assurance mechanisms. Out of the 15 answers, 7 (46.6%) stated other and out of these 7 others, 4 (57.1%) answered that they do not verify third-party extensions at this moment or yet.

Opening up a software platform usually indicates that extensions will increase. The results show that there is no favorable phase in the life cycle of when an extension is to be added. Although most platform extensions (26,1%) are implemented in the commissioning/deployment phase. 21,7% of the participants extensions are integrated regularly and multiple times per release, 21,7% also have an explicit integration phase for each release. The least common practice is for end-users to add extensions directly to the running system.

The use of different strategies showcase that there is no definitive strategy. All projects have prioritized different aspects and put emphasis on the parts they thought would mostly benefit their product and company as a whole.

4.3 Technological aspects

To build software platforms you need a programming language. Our questionnaire respondents answered which they had used to realize their platforms and the most common languages were; C used 6 times (40.0%), C++ used 8 times (53.3%), Java used 5 times (33.3%) and C# used 4 times (26.6%). The option "Other::" got 4 different additional answers that were used in combination with one of the aforementioned programming languages where one was left blank (Not answered). These being HTML/CSS/JS 1 (6.6%), Domain specific language 1 (6.6%), JS 1 (6.6%) and Not answered 1 (6.6%). Out of the 15 participants that had answered in full, 8 (53.3%) answered that they used more than 1 language in their project.

Out of the 15 participants that completed the questionnaire, 2 respondent chose not to answer what kind of extension mechanism they had used in their software platform. The percentages are calculated using 13 answers in total. The most used extension mechanism was the use of APIs at 76.9%, making it the most used extension mechanism with a plug-in system being the second most used with 53.8%. Thirdly comes the use of Isolated run-time containers and Web-services used in 5 (38.5%) platforms. Used in 4 (30.8%) platforms is Domain specific languages (DSL) and least used in our respondents platforms is Explicitly formulated conventions which was used in 3 (23.1%) platforms.

Out of the 15 respondents who completed the ques-

Aspect that needed change	Str. agree	Agree	Neutral	Disagree	Str. disagree
Business model	4(28.6%)	5(35.7%)	2(14.3%)	2(14.3%)	1(7.1%)
Platform architecture	3(20.0%)	6(40.0%)	4(26.6%)	2(13.3%)	0(0%)
Platform development process	3(20.0%)	8(53.3%)	4(26.6%)	0(0%)	0(0%)
Organization of the development	2(13.3%)	7(46.6%)	6(40.0%)	0(0%)	0(0%)

Table 3: Aspects that needed change.

Programming language	Extension mechanism	Control of execution
C#	API, Web Service, Plug-in, EFC	P executes E
C++	API, Web Service, Plug-in	P executes E
C++, C, Java	API, Web Service, IRC	P executes E
C++, C	API, EFC	P executes E
C	API, Plug-in, IRC, EFC	P executes E
C, Java, JS	API, DSL, Plug-in, IRC	P executes E
C#, C, C++, DSL	API	E executes P
C++, Java	IRC	E executes P
C++, Java	Web Service, IRC	E executes P
C++	API, DSL, Plug-in	Other:: Both
Java, CSS/HTML/JS	API, Plug-in	Other::Both
Other	API, Web Service, DSL	Other::Both
C#	N/A	N/A
C#	N/A	N/A
C++, C	Plugins	Other::Unclear question.

Table 4: Programming language, extension mechanism and control of platform execution for the projects.

tionnaire, 2 did not answer which option of execution they use. The execution of the deployed platform control is split between three ways. Option one is having the extensions execute the platform, option two is having the platform execute the extensions (inversion of control principle). The third option was Other where 3 out 4 respondents answered that both alternatives are used depending on the extension point. The last respondent did not fully understand the question. Having the platform execute the extensions, option two, is the most used option being used in 6 (46.1%) out of the 13 projects, and is used twice as often as option one and three which are used in 3 (23.1%) out of 13 projects. The results of three main categories are summarized in table 4.

When asked how they support the development of extensions the participants show that the development of extensions are well supported in most projects. The documentation and support for the development of extensions greatly increases the chance that external developers can aid and innovate the development of the software platform and help with specialized requirements. Out of the 15 participants that completed the questionnaire, 14 (93.3%) utilize Interface/API-Documentation in their projects, 12 (80.0%) utilize Tutorials/How-to's and Code examples, 11 (73.3%) utilize Development guidelines, 6 (40%) have a Software development kit (SDK), 5 (33.3%) utilize code templates and 2 (13.3%) projects use other means of supporting the development of extensions.

4.4 Interview results

To strengthen the questionnaire results and to aid us in answering our research questions, interviews were

held with two industrial participants. This section is a summary of the two interviews held and will be discussed in the next part to strengthen the conclusions drawn from the questionnaire result.

4.5 Interview results of an open redesigned platform

Company X: Their software platform was a complete re-design. They originally tried to open an internal product but quickly discontinued their attempt and began redesigning the platform. The key reasons for their redesign was that they wanted to reduce cost integration, make it more reusable by external customers, increase fault reporting and in particular redesigning the API/Plug-and-play.

They faced some challenges when redesigning their software platform. The customer wanting it to be made using C# was one struggle they faced. A significant change to the Domain model also led to consequences that required them to change the API. Changing the API also meant having to support the legacy API that was redesigned 20 years prior. Another struggle was the protection of intellectual property. Making the code clean and placing functionality at the most efficient places meant giving competitors access to parts of the code that was undesirable for them to share. This led to the software architect having to find a trade-off between efficiency and placement of code. They learned from these challenges were that separating the API and removing the legacy API should have been done a lot sooner, to remove the hassle of having to support an old incompatible version for only legacy users. They also realized that they need to be more precise and clear in their communication as well as documentation so that customers expectations match what the platform can provide.

The use of communities and building up one was considered less important to them because they tend to have one-to-one communication with their customers since their customers more often than not, are competitors who you might not want to share your secrets with. As a part of the software platform being open, customers are given the ability to test an evaluation version. This evaluation platform is thought of as a means for users to test before purchasing. For the original platform there is a full release every six months. However, for the latest version there is a patch release every 3 weeks.

One of the key elements to this software platforms success was dedicated towards reliability. The interviewee praised reliability to the extent that it is considered one of the major reasons to why it surpassed a strong competitor. Another element discussed is the domain knowledge that has been obtained over the years. The company considers domain knowledge to be a must for a software platform to survive and compete in the market.

4.6 Interview results of an open cloud-based platform

Company Z moved all their software to open source projects creating a cloud based platform. The users of the platform are developers who can create their own services and deploy them into the cloud. These services can then be used by other services already running in the cloud infrastructure, making it a compositional approach. It was planned to have different data centers where the cloud infrastructure runs but for now there is one data center that is the central space. The first iteration of the opening up of the platform was creating something called suits where several tools and connectors between tools are contained. The next step was to utilize these tools and build up a new micro service which was then used in the project cloud. The micro services in the cloud infrastructure uses strings, string roots and the old string ecosystem. In the infrastructure there are different data stores that run on several data bases. Interfaces are described by the means of rest API's. Using the old tools functionality the micro services were created and those were then connected with the rest API's.

The reasons for opening up were competition, compatibility, and difficulties maintaining the platform. In the beginning the company had different tools

from different locations, this meant that a lot of bridges were needed so the decision was made to connect those different tools. Before some parts were C++ and some where Java and there was the need to connect them via calls so rest API's made it much easier. Another big issue for company Z was that the old tools were monolithic in their architecture.

This platform's underlying technology is a cloud infrastructure where you can deploy new services, these services are called droplets. A droplet is similar to an Android app and has a life cycle. This underlying technology has applications inside which runs the cloud infrastructure. These droplets are the means of creating a new functionality by creating and deploying a service. The droplet is deployed into the cloud infrastructure and it knows the domain name of other services in the cloud, which it then calls using Rest API. The communication between the droplet and the Rest API improved the integration of components.

The functionality was extracted from the old tools and added to the Rest APIs making them exist in the cloud. Although the old tools are capable of more than those that are now available in the cloud, it is important for company Z to open up the platform so buyers can create services, thus showing the company what users want. The code of the platform has the same base, but it was rebuilt in order to make it a cloud service. Everything in the interface part is new but the core business logic is the same. Company Z does not allow bad interfaces to exist in their platform, and the good ones have had several iterative improvements.

To analyze the legacy code company Z had a strategy where they determined what functionality to offer. Next they analyzed the resources available and how these resource could be mapped to the platform. Then they performed some iterations, such as writing example applications. One exercise that they performed were hackatons where developers were asked to use the services in the platform and create their own services. This showed them what could be improved and what was missing.

The lessons learned were that there was a lot of effort in opening up a platform, it is time consuming and you do not have to rewrite everything but instead you can reuse parts of the old tools. One unexpected lesson learned was that they had expected more from the underlying technology, making them have to develop a lot by themselves.

5 Discussion

The questionnaire results provided more questions than answers, since this study applies grounded theory those questions could be investigated further by using interviews. This study is aimed towards assisting in creating guidelines on how to open up a software platform in an industry. There is not enough data support to draw the guidelines but there is enough data to discuss what type of guidelines could be considered. Discussing the different results from the survey, common traits that the different software platforms have are discussed. The results also showed underlying reasons that industries have for opening up a software platform. This gave us the ability to do selective coding; creating relationships between main categories.

5.1 Strategies and plans for opening up a software platform

The use of strategies and a planned process is often a crucial step when developing software or any product in general. Developing a strategy gives you something to follow and makes it easier to predict risks that could arise at later stages. Being able to predict how something can turn out, will aid efforts in planning for these risks. The questionnaire showed that most projects needed to somehow significantly change their architecture, their development process, their development teams as well as their business plan and this is far from desirable when developing your product. Anvaari and Jansen (2010), suggest that by looking at how open a software platform is, could potentially give insight into architectural factors that show "the openness strategy". Looking further into how open

the different platforms were could perhaps have garnered us a better understanding of architectural factors that affect the outcome when opening up a software platform. Better planning of a software projects architecture for a software platform seems crucial and should be researched more to help form guidelines. Guidelines of how to best plan your software architecture in a way that warrants little to no changes when the project has been deployed. From the first interview, company X idealized domain knowledge and thought that one could/should not open a software platform without domain knowledge. Henk, Slinger and Sjaak's (2011) study supports the importance of domain knowledge in a closed platform. It is now also possible that domain knowledge is just as important for open software platforms and the process of opening a software platform. In our opinion this is one of the more important factors when opening up a software platform.

Company Z had a quality assurance strategy, where they had teams, made up of both developers and administrators, for every provided service in the platform. This means that there is a team actively checking new services added by users. This strategy could be suited for a platform where the expected number of new services created by users are known to be quite few but vital. Another strategy that company Z had was to test the platform by allowing a set of users, in this case developers, to play with the platform in order to understand what needs to be evolved further or what is lacking in the platform. Testing before and during the process of opening up a software platform could remove potential threats.

5.2 Decisions and reasons for opening up software platforms

From the results we can see that the decision to open a software platform is generally the realization that innovation exists both internally and externally. As well as realizing that certain requirements and functions would be too difficult for the employees to provide. The results of this study has not shown a single duplicate of the benefits gained. Not being able to predict the benefits could be considered an uncertainty that makes industries hesitant to open a software platform. From the results we can also see that an increase in users is a common reason for industries to open up their software platform. Stakeholders are vital to an industry and current stakeholders will have a tendency to refuse any change. One needs to remember, as B. Balter (2015) mentions, that these stakeholders are internal, being more open will allow the attraction of external stakeholders[6].

All of the participants had problems with their closed platform. An overflow of requirements from users and market competition were the biggest issues with the closed platforms. Another common reason to open up a software platform within an industrial setting is innovation. Considering that strong software is now days being considered a necessity to achieve quality and loyalty, the demand to improve software is constant. According to the results, software platforms similar to the Android Os is the type that industries most commonly open up. Perhaps because this type of software is the easiest to open up? Within the industrial world open software platforms are targeted towards users that are departments and developers within an organization. Allowing employees and departments to give input on a platform which they use every day aids companies in finding innovation through internal development. Perhaps another reason to why these particular users are targeted could be that an industry software platform commonly poses safety issues regarding intellectual property. As Company X explained, the protection of intellectual property becomes a trade-off point where you either allow competitors access to undesirable parts and have clean code or restrict access and have inefficient code. This gives the option of not allowing complete openness to the public but a controlled openness for the company in question.

5.3 Sustaining the success of an open software platform

Extension mechanisms are a great tool that gives external developers a chance to aid in an organizations

development. In our questionnaire, the most used extension mechanism was API's, which was utilized by 76,7% of the projects. The research by Kilamo et al. (2011), state that companies that have gone through the process of opening up proprietary software, learned that more effort should have been put towards; re-factoring the code for external contributors and building a community.

Most questionnaire respondents added that they utilize some form of tutorials or code examples to aid external developers. This correlates to the fact that 100% of all questionnaire respondents strongly agree or agree that the software quality of the platform is an important aspect. 92.9% of all questionnaire respondents strongly agree or agree that the software quality of the extensions is also a very important aspect of sustaining the open software platforms success. Comparing to other aspects of success for opening up a software platform, providing sufficient guides for customers is little effort that could be greatly beneficial. From the interview of Company X it was noted that the aid for external developers was not just one alternative but almost every alternative there exists.

From company X we learned that they did not have a community setup in order to aid the software platforms growth since most customers are competitors and the interactions were one-to-one based. In the research by Kilamo et al. (2011), a company explained that after evaluating their opening process and use of a framework, they would have put more effort into the community aspect to make it self-sustainable. When sustaining the success it is important to acknowledge that an open software platform is a living organism within an organization and its health is important. Jansen's(2014) model of operationalization can determine the health of an ecosystem. Determining the health of the software ecosystem will provide insight into which areas of the software platform needs to be altered in order for a healthier ecosystem. Since a platform is an underlying basis for a software ecosystem, keeping an ecosystem healthy would result in sustaining the success of an open platform. An open software platform needs to be constantly maintained and improved otherwise competition might strangle it. Sustaining the success is a considerable effort that a company needs to undertake and partially aid in answering SQ2.

5.4 Technological aspects and challenges when opening up a software platform

The most used programming languages among the participants responses are; C++, C#, Java and C. There is evidence in our result that what kind of extension was used stem from what type of programming language is utilized but it is not concrete enough to be able to draw a supported conclusion to which is best. Out of the 13 fully completed answers (2 did not answer), there are a total of 6 different types of extension mechanisms used. Some extension mechanisms are used in the same project and the most commonly used ones are API's and Plug-ins. The type of extension mechanism used does seem to correlate to what kind of execution control is used but the limited data makes it hard to say for certain. The execution of the deployed platform control consist of 2 different ways of execution and could depend more on the type of platform than how the platform is designed. Whether one way of control is more crucial to an open platform is hard to determine and would need further investigation. What we can see from table 2 is that you can use a variety of extension mechanism as well as programming languages, and the control of how you execute these extensions might not always correlate. In all the projects where the platform executed the extensions, API's are used, which includes the projects where both ways of execution is used.

As could be seen in the results, the most common challenges that arise during the opening of a software platform have to do with technology. Restructuring the architecture, introducing new technologies, and being able to maintain backwards compatibility. These all cause trouble for projects due to the lack of knowledge surrounding the process of opening a software platform. When opening up a software platform, one of the main strategies is to re-use several parts of the old platform. This is a good approach but one needs to remember that with opening up there

comes new aspects to the technology that needs to be considered. Company Z could only use subsets of their old tools. New connections had to be created and new user friendly interfaces had to be made from scratch. When opening up a software platform, it is vital to remember that the platform is no longer manipulated from one side, but from multiple sides and these sides have to communicate without disturbing each other.

6 Threats to validity

Since the questionnaire had almost 100 partial responses which were excluded from the study, there is a potential internal threat that majority of the participants were not appropriate for this study. Although some of the software platforms were not completely opened yet, their data was still considered. This could be considered an internal threat but the data was still important to the study since it included information that could help draw conclusions of why a software platform is opened. The data analysis was done by a human being, and what one person deems as interesting might not be interesting in another persons eyes. To avoid this internal threat the analysis and its results were done in a pair-wise manner. Every result, discussion and conclusion was made in a pair-wise way manner, meaning the content has been written, discussed and then re-written more than once. Related work guided and supported many of the conclusions, this could be a internal threat since there are many research studies out there that do contradict each other.

The questionnaire responses account for the majority of data, However, since there were such a vast number of partial responses there is an external threat to the quality of the questions. There is an uncertainty to why there were so many partial responses, the questionnaire could have been too long, the questions could have been too complex, the questions could have posed a security threat, or the participants that were asked to join had no valuable input to the study. From the excluded interview it is clear that there is an uncertainty as to weather the product in question is an open software platform or not. If this misunderstanding is common then it poses a construct threat to the results and conclusions of this study.

7 Future work

Taking into consideration all the areas discussed, we believe that the areas that need more research and could be explored further are technological aspects. Why these are chosen and how they correlate to each other. From the results we can see that they do not depend on each other and might have more to do with how the software architecture was initially designed but it is unclear from the questions asked and would therefor require more research for a deeper understanding of why these are chosen. Perhaps they are chosen because of how the software platform is intended to work. Eisenmann et al. (2008), looked at factors that contribute to the decision of the opening or closing of a software platform. They concluded that openness depends on where you allow access for outside contributions. Which also suggest that by looking at where a software platform is opened for external developers could have been useful when trying to understand which factors, other than the technological, affect the outcome of opening a software platform, better answering RQ1. Another area of interest would be strategies used, the software architecture and how they are initially designed. Getting a better understanding of how software platforms are designed to be open or restructured might give better insight into how projects should be designed. This could have answered questions like if a company should open up a closed platform or design it to be an open platform initially. Licensing and legal concerns have not been discussed in this thesis but is however aspects that could affect the outcome when opening up a platform. A company would perhaps need to look over trade-offs that affect the intellectual property of the company and efficiency of the platform.

8 Conclusion

The below parts summarize aspects that answer our research questions. Paragraph one discusses what the study revealed as important aspects for a successful opening of a software platform within an industrial setting. While paragraph 2 and 3 discusses what leads a company to open up their software platform and how it affected the company after opening it. The last paragraph contains the conclusions we drew about the technical aspects that are necessary when one opens a software platform. These technical aspects could be used as a guideline if uncertain on how to plan your software platform to be opened up. When an industry plans to open up a software platform we believe that strategy and domain knowledge go hand-in-hand.

From the first interview, company X idealized domain knowledge and thought that one could/should not open a software platform without domain knowledge. Henk, Slinger and Sjaak's(2011) study supports the importance of domain knowledge in a closed platform. From the study it is now also possible to conclude that it is just as important for successfully opening up a software platform. There are several important aspects to consider when opening up a previously closed software platform. Before opening up it is important to discuss, determine, and understand the underlying technologies that will be used. The study has revealed that it is possible for companies to expect too much of technology that is introduced to the development. We can also conclude that designing or redesigning a software platforms architecture is detrimental to any project that wants to develop an open software platform. As to not design a software platform that is cost inefficient or time consuming when having to restructure and redesign it.

There is a significant relationship between the industries reasons for opening up a closed platform. It is revealed that the main struggles with closed platforms are due to an overflow of requirements, intense market competition and lack of compatibility with other platforms. These problems alone would not push the opening of a software platform. However, adding the industries business and technical intentions would. These three are the common factors that push an industry to open up their closed platform.

Companies were affected differently. Benefits and drawbacks that companies had varied to such an extent that it is impossible to validate what is common. It can be speculated that the reason is that there are no guidelines on how to open up a software platform, meaning that the process of opening up differs between the companies. However, different domain specific platforms using the same process of opening up could still end up having different results. Even if drawbacks and benefits are different, the way that companies have to sustain the success after opening up their software platform is fairly similar. Other factors of sustaining the success includes means to able external development, maintaining the quality of the extension mechanisms and the software platform, keeping the extensions stable, and managing the community.

From the study we can not conclude what technological aspects should be used, but what aspects are most common. The study showed that there is no clear correlation between aspects like programming language, extension mechanisms used and how the execution of the deployed platform is controlled. What we can determine is that using a programming language that gives structure is key and that which one is determined by what you need out of the language for your project. The most common way to execute your deployed platform is having the platform execute the extensions. The type of extension mechanism most commonly used with this is API's. Which is used in all projects where the platform executes the extensions. Using a plug-in system or a Web Service is also quite common.

9 References

[1] A. Gawer, The organization of platform leadership: an empirical investigation of intel's management processes aimed at fostering complementary in-

novation by third parties, PhD Thesis, Massachusetts Institute of Technology, 2000.

[2] A. Gawer and M Cusumano, Driving High-Tech Innovation: The Four Levers of Platform Leadership, 2001.

[3] A. Gawer and M Cusumano, Platform Leadership How Intel, Microsoft, and Cisco Drive Industry Innovation, 2002.

[4] A. Strauss and J. Corbin, Grounded Theory Methodology, In Handbook of Qualitative Research, 1994

[5] A. Strauss and J. Corbin, Grounded theory research: Procedures, canons, and evaluative criteria, Qualitative sociology, 1990.

[6] B. Balter, Five best practices in open source: internal collaboration, 2015.

[7] C. RB. de Souza, et al., The Social Side of Software Platform Ecosystems, Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ACM, 2016.

[8] D. Hilkert, A. Benlian, and T. Hess, The Openness of Smartphone Software Platforms – A Framework and Preliminary Empirical Findings from the Developers' Perspective, Annual Conference of the Gesellschaft für Informatik, 2011.

[9] D. Hilkert, et al., Perceived Software Platform Openness: The Scale and its Impact on Developer Satisfaction, 2011.

[10] H. Schuur, S. Jansen and S. Brinkkemper, The Power of Propagation: On the Role of Software Operation Knowledge within Software Ecosystems, Department of Information and Computing Sciences, Utrecht University, 2011.

[11] J. Bosch, From Software Product Lines to Software Ecosystems , 2009.

[12] K. Armas, The search for Meanings-Grounded Theory, 1996.

[13] K. Boudreau, Open Platform Strategies and Innovation:Granting Access versus Devolving Control, Management Science, Volume 56, Issue 10, 2010.

[14] M. Anvaari and S. Jansen - Evaluating architectural openness in mobile software platforms 2010.

[15] N. Pandit, The Creation of Theory: A Recent Application of the Grounded Theory Method, University of Manchester, 1996.

[16] R. Costanza and M. Mageau, What is a healthy ecosystem?, 1999.

[17] S. Jansen, Measuring the Health of Open Software Ecosystems: Moving Beyond the Scope of Project Health, Information and Software Technology Journal, special issue on Software Ecosystems, 2014.

[18] S. Jansen, A. Finkelstein, and S. Brinkkemper. A sense of community: A research agenda for software ecosystems. 2009.

[19] S. Jansen, and M. Cusumano, Defining Software Ecosystems: A Survey of Software Platforms and Business Network Governance, 2012.

[20] T. R. Eisenmann, G. Parker and M. Alstyne, Opening Platforms: How, When and Why?, Harvard Business School, 2008.

[21] T.Kilamo et al., From proprietary to open source—Growing an open source ecosystem, Journal of Systems and Software, Volume 85, Issue 7, 2012.

Appendix: Survey on Opening Software Platforms

General Properties of the Open Platform

1. What is the name of your platform?
2. What is the domain of your platform (e.g., finance, software development, games, content management)?
3. If you had to characterize your open platform by comparing it to other well-known platforms, to which extent do you agree it is similar to one of the following platforms?
4. Who are the users of your platform?
5. Using which of the following programming language(s) is your platform realized?
6. What are the extensions to your platform called?
7. How large is your platform currently in lines of code (LOC)?
8. How many people are currently and actively involved in developing, maintaining, and testing the platform?

Rationales for Opening Up the Platform

9. Were there any problems with the closed platforms that led to opening it up?
10. What were the business intentions for opening up the platform?
11. What were the technical intentions for opening up the platform?

Platform Opening Process

12. How long did the platform exist before it was opened?
13. How long did it take to open up the platform? Leave empty if still ongoing
14. What was the starting point of the opening process?
15. How many people were/are actively involved in opening up the platform?
16. Can you briefly describe the process or strategy you followed when opening the platform?
17. How is the execution of the deployed platform controlled?
18. Which of the following extension mechanisms did you incorporate to open the platform and which technology was used?
19. For opening the platform, did you need to significantly change one of the following aspects?
20. Did you face any particular challenges when opening-up the platform? If so, where?

Consequences and Success Criteria

21. Considering the entire process of opening up the software platform, to which extent do you
22. How did the size of the code base change as result of the platform opening process?
23. What were the particular benefits of opening-up the platform?
24. What were the particular drawbacks after opening-up the platform?

Sustaining the Open Platform

25. Which of the following aspects do you find very important for sustaining your open platform?
26. How do you support the development of extensions?
27. Which mechanisms do you use to verify the quality of third-party/external extensions?

28. How many extensions exist for the open platform (apps, plug-ins, components etc.)?
29. At which stage of the platform lifecycle does the platform first get in contact with an extension?

Role and Personal Details

30. What have been your roles in software-platform development?
31. How many years of industrial experience do you have in software Engineering?

Contact

32. Do you want to receive a report with the results of the study?
33. May we contact you with clarification questions on your answers?

Final Remarks

34. If you are willing to be contacted for further clarification questions or to receive the study results, please give us your name and email address.
35. Are there any final remarks you would like to tell us (e.g., if you canceled the opening process or the procedure failed, please give reasons)?