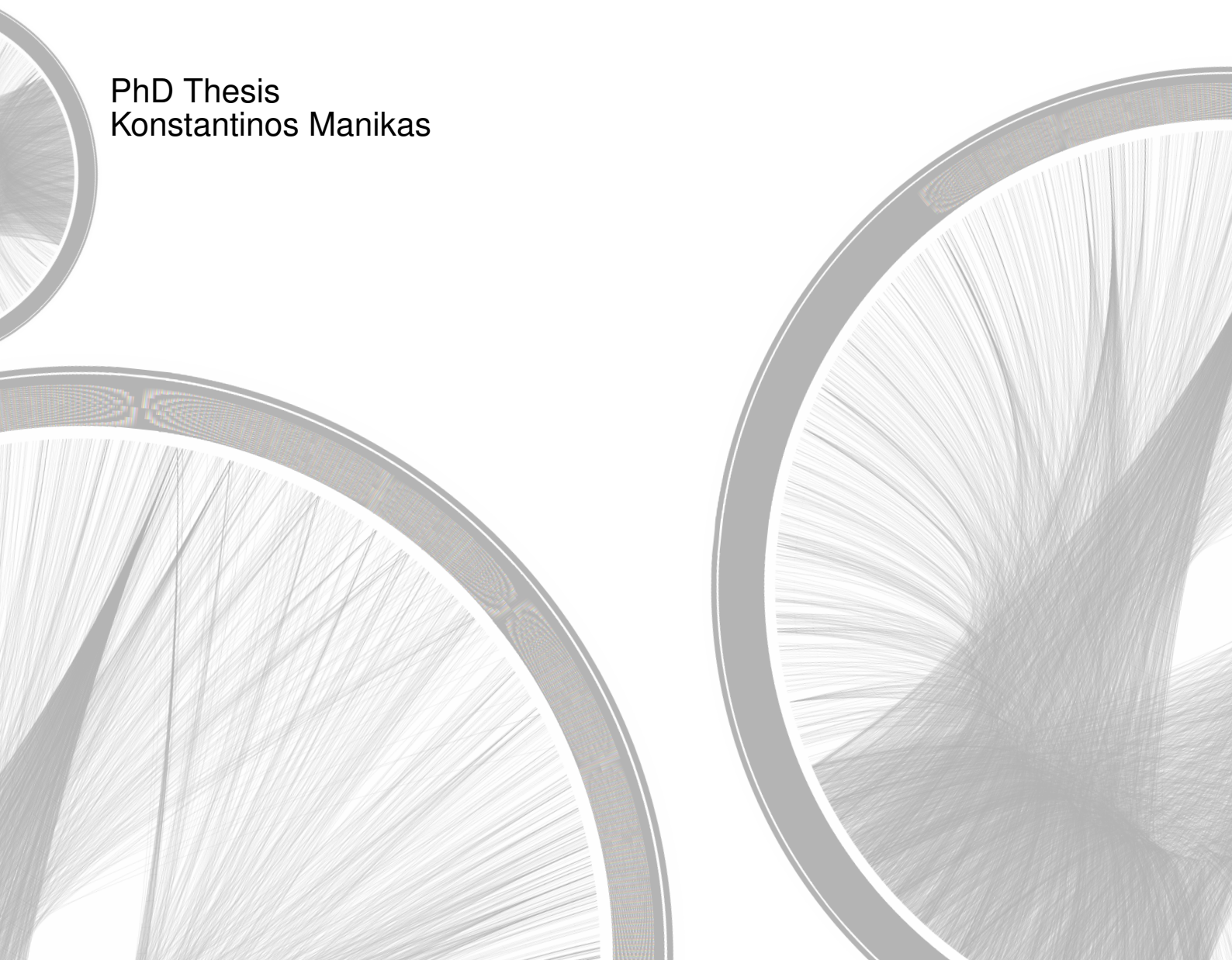
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Analyzing, Modelling, and Designing Software Ecosystems

Towards the Danish Telemedicine Software Ecosystem

PhD Thesis
Konstantinos Manikas



PhD dissertation
Author:
Konstantinos Manikas
Human-Center Computing
Department of Computer Science
University of Copenhagen
Denmark
Title: Analyzing, Modelling, and Designing Software Ecosystems – Towards the Danish Telemedicine Software Ecosystem

Academic advisor: Klaus Marius Hansen

PhD evaluation committee:

Slinger Jansen
Assistant Professor
Department of Information and Computer Science
Utrecht University
Netherlands

Yvonne Dittrich
Associate Professor
Software Development Group
IT University of Copenhagen
Denmark

Finn Kensing (chairman)
Professor
Center for IT Innovation (CITI)
University of Copenhagen
Denmark

Typeset: LaTeX
Cover page graphics: Apache Felix component connections depicted in Circos (`http://circos.ca/`)

Abstract

As in many western countries, Denmark's demographic changes and the organizational and financial changes of the Danish healthcare are challenging the levels of provided healthcare and point towards the establishment of telemedicine as means of patient support, diagnosis, and treatment. However, telemedical solutions, despite the benefits they provide, are faced with a number of challenges that inhibit their adoption, development, and implementation. In this project, we examine the approach of a software ecosystem as means of addressing these issues. A software ecosystem can be roughly explained as the software development and distribution by a set of actors dependent on each other and the ecosystem. We commence on the hypothesis that the establishment of a software ecosystem on the telemedicine services of Denmark would address these issues and investigate how a software ecosystem can foster the development, implementation, and use of telemedicine services.

We initially expand the theory of software ecosystems by contributing to the definition and understanding of software ecosystems, providing means of analyzing existing and designing new ecosystems, and defining and measuring the qualities of software ecosystems. We use these contributions to design a software ecosystem in the telemedicine services of Denmark with (i) a common platform that supports and promotes development from different actors, (ii) high software interaction, (iii) strong social network of actors, (iv) robust business structures, supporting actor involvement in the ecosystem, and (v) proper orchestration and governance of the ecosystem to promote and support the changes and the health of the ecosystem.

Our work contributes to Net4Care, a platform to serve as the common platform in the software ecosystem under establishment. In addition, it contributes by providing input and guidelines on the role and activity of 4S organization, an organization to serve as an orchestrator in the ecosystem with the aim of managing the platform, supporting actor and software interactions, and promoting the ecosystem health. This thesis documents the groundwork towards addressing the challenges faced by telemedical technologies today and establishing telemedicine as a means of patient diagnosis and treatment. Furthermore, it serves as an empirical example of designing a software ecosystem.

Dansk resumé

Danmark står som så mange andre vestlige lande over for udfordringer i levering af sundshedsydelser. Dette taler for etableringen af telemedicinløsninger, som en måde til at understøtte patienterne i at blive mere selvhjulpne, i diagnosticering og i behandling. Telemedicinområdet står dog også over for en række udfordringer som hindrer indførelse, udvikling og implementering. I dette projekt undersøger vi "software-økosystemer", som en tilgang til at addressere disse udfordringer. Software-økosystemer består af en software-platform og indbyrdes afhængige aktører som udvikler og distribuerer software bygget oven på platformen. Projektets hypotese er, at etableringen af et software-økosystem inden for danske telemedicin vil kunne adressere en række af problemstillingerne og projektet undersøger, hvordan et software-økosystem kan understøtte udvikling, implementering samt brug af telemedicinydelser.

Vi analyserer eksisterende software-økosystem teori og udvider den ved at definere begrebet "software-økosystemer", bidrager med teknikker til at analysere eksisterende økosystemer, teknikker til at designe nye økosystemer samt måder til at definere og måle kvaliteten af software-økosystemer. Vi bruger efterfølgende disse bidrag til at designe et software-økosystem inden for telemedicinområdet, hvor software-økosystemet har (i) en fælles platform der understøtter at forskellige aktører udvikler software ovenpå, (ii) en høj grad af software-interaktion, (iii) et stærkt aktørnetværk, (iv) en robust forretningsstruktur, som understøtter aktørernes engagement og tilknytning til økosystemet og endelig (v) en orkestrering og styring af økosystemet, der tillader og understøtter dets dynamik og sundhedstilstand.

Vores forskning bidrager også til Net4Care, en platform der virker som den fælles integrationsplatform i det software-økosystem, som vi har designet. Ydermere bidrager vi med input og retningslinjer for roller og aktiviteter i organisationen der nu udvikler Net4Care, 4S. 4S tjener som en "organisator" i økosystemet med det formål at håndtere, styre og drive platformen, fremme aktør- og software-interaktion samt at fremme økosystemets sundhedstilstand. Denne afhandling dokumenterer arbejde der grundlæggende adresserer de udfordringer, som telemedicinområdet står over for i dag. Endelig tjener afhandlingen som et empirisk eksempel på et design af et software-økosystem.

Acknowledgments

Copenhagen, November 2014

The title of Doctor of Philosophy is an academic title appointed as acknowledgment of scholarly contributions. Although this title is personal, there can be people who support and assist one to obtain this title. In this journey I have been of luck to associate with some exceptional people that have contributed to my efforts to complete this project.

First of all I would like to thank Sonatype for providing the configuration files to conduct our analysis in Paper 2. Axis and their partners for sharing their views in Paper 4. The Capital Healthcare Region of Denmark and Judith Lørup Rindum for providing the list of telemedicine applications used in Paper 5. The people of the Software Engineering Research Group of Lund Technical University in Sweden for welcoming me and being good hosts. Especially Per Runeson for his insightful comments and support and Krzysztof Wnuk for the good collaboration and warm fellowship.

I would like to acknowledge the work of all who participated in the Net4Care and Connect2Care projects. More specifically, I would like to thank Henrik Bærbak Christensen for our collaboration, good times, and showing that the answer is "42". Furthermore, I would like to thank Morten Kyng from the 4S organization for the collaboration and support. I would like to express my gratitude to my academic advisor Klaus Marius Hansen, for being a real mentor and especially for being reflective, sincere, and always finding time. I would also like to thank my colleagues in the Human-Centered Computing for creating a nice working climate and providing nice feedback in this work.

Furthermore, there are a number of people that contributed to this work indirectly by supporting me personally and making life easier. I would like to thank "mormor" Jytte and "morfar" Benny for their love and support. The family in Greece for loving me, tapping me in the back or comforting whenever I needed it. Vaso and Nikos for their immense love, making them to go out of their way to support me. Maria Ie, for her love, support, and being a real companion in this journey making every challenge a learning point. Aenias, for his patience, for making this journey so pleasant, and for all the put-to-bed times that I was forced to slow down, take a deep breath and reflect on aspects of this project.

From the bottom of my heart: Thank you. This journey might have not been possible and it would definitely not be half as fun without your influence.

Contents

Part I

Summary

Chapter 1

Introduction

Healthcare in Denmark has undergone a number of changes related to the demographics of the country, organization, and funding. These changes, as explained below, are pointing to the adoption of telemedicine, i.e. the provision of healthcare at a distance, as means of addressing the increasing demands in better healthcare quality at lower costs. The recent demographic challenges in Denmark, following the trend of the western countries, affect the services of care. The two patient groups that are in most need for constant healthcare services, elderly and patients with chronic diseases, are growing. In 2011, one out of six persons in Denmark was above 65 years (Francesca et al., 2011) while this number is estimated to increase by 60% in 2044 (Danmark Statistik, 2011). Moreover, one out of three persons in Denmark is suffering from at least one chronic disease, while this number increases to more than half of the population of age over 75 (Christensen et al., 2011). These numbers are challenging the economics of a tax-funded public healthcare, considering that life expectancy of newborns in Denmark has increased by three years since 2000 while births have decreased by almost 20% for the same time[1]. Moreover healthcare services in Denmark are expensive with costs reaching up to 11% of the country's Gross Domestic Product (GDP) (OECD, 2014) where 2% of GDP is used for long term care alone in 2007 (Francesca et al., 2011).

The need for restructuring healthcare services towards the use of telemedicine is also supported by the constant increase of patient-hospital distance. This has been the case in the last 15 years where hospital administration and management departments, previously part of each hospital, have merged covering up to whole geographical regions (Kristensen et al., 2008), following the overall general healthcare centralization trend with examples in England, Norway and Sweden (Fulop et al., 2002; Magnussen et al., 2007; Ahgren, 2008). This centralization approach was also formally established in the organization and financing of care with the 2007 reform where specialized services and equipment were centralized in few and bigger hospitals, while primary diagnosis and care was decentralized and conducted to a wider extent by clinics, day hospitals or general practitioners (GPs)(Olejaz et al., 2012; Danske Regioner, 2007).

A possible restructure towards incorporating telemedical technologies into

[1]The statistics are calculated from Statistics Denmark (2014) and can be seen in `http://www.diku.dk/~kmanikas/ds/lifeexpectancy/` and `http://www.diku.dk/~kmanikas/ds/birthrate/` respectively.

the healthcare services would also require a level of technological maturity. The development of telemedicine systems today is supported by a set of standards and systems both in international level, with standards like HL7, CDA, PHMR, XDS.b (HL7, 2014, 2010; IHE, 2013), and in Danish level with systems like the "Shared Medication Record" (FMK) (SSI, 2014a), "Danish Healthcare Data Network" (SDN) (MedCom, 2014a), "Danish Healthcare Video Hub" (VDX) (MedCom, 2014b), "National Service Platform" (NSP)(SSI, 2014b), or sundhed.dk (Sundhed.dk, 2014).

At the same time, the use of telemedicine technologies from patients and clinicians has never been easier. On one side people today are highly connected to the internet, with close to 70% of the people in Denmark between 55 and 74 years regularly accessing the internet (Rodrigues et al., 2012). While, on the other side, the advent of the smart devices has brought us closer to a digitalized every-day life, where at any point we can upload pictures to social media, read news, and chat. This digitalization of everyday life has also entered the healthcare with the Apple AppStore counting more than 40,000 apps under the category "health & fitness" (Apple App Store, 2013) and trends like the "Quantified Self", where people collect information about their body functions and habits to help them, among others improve a health condition, increase sports performance, or live healthier (Quantified Self, 2014; The Economist, 2014).

However, telemedicine in Denmark, although the accepted benefits it provides, only counts sparse solutions instead of being a countrywide established means of patient support, diagnosis, and treatment. Telemedicine solutions in Denmark are faced with a number of challenges that stand in the way of development and implementation of new solutions. These challenges, as explained below, can be summarized as the fragmentation of healthcare services, fragmentation of healthcare systems, and high level of proprietary solutions.

Danish healthcare is divided into different administrative (and funding) organs following, to a big extent, federal structure. This administrative structure allows each of the administrative organs, e.g. the five healthcare regions, to prioritize and organize, within their area of responsibility, how the services are provided, while providing an obstacle on countrywide policies applied. This structure is also reflected in the structure of Information Technology (IT) systems. This is demonstrated, in the regional level, by the existence of four different Electronic medical Records (EMRs) (Kierkegaard, 2013), while a unified national EMR "is not likely in the near future" (Rudkjøbing et al., 2012). The fragmentation of the healthcare organization and IT systems are obstacles in the expansion and adoption of telemedicine systems on a wider scale. From one side actors developing telemedicine systems might need to be introduced and accepted from each federal separately, while from the other side, the difference in IT systems makes the integration with new systems challenging and resource demanding.

Moreover, IT systems in healthcare, and more specifically in telemedicine, have the tendency of having local (and duplicate) implementations of solutions to similar problems (Manikas and Hansen, 2013a). This tendency is also noted in other IT initiatives in the healthcare where systems are fragmented, characterized by proprietary solutions that are prone to close other systems out, not allowing interoperability. An example is the lack of one common electronic medical record mentioned above (Rudkjøbing et al., 2012). This was also pointed out by Delloitte (2007), in their study commissioned by the state in 2007, identi-

fying that healthcare data is hard to be reused while underlining the importance of potential reuse.

Issues like the above mentioned pose an obstacle to the development of new telemedicine solutions and put a burden on the already expensive history of healthcare project failure with GEPJ and NPI (Kierkegaard, 2013; Aanestad and Jensen, 2011; Lund, 2013) impacting the economy of a public healthcare but also bringing political consequences by affecting the public opinion negatively.

Different software domains have recently incorporated a method of developing and distributing software by opening up (or establishing) a common platform (or technical infrastructure) in a way that allows different actors to contribute on top of the platform and results in a set of products or services. The different actors (internal or external to the platform) and the software produced are characterized by the range of symbiotic relationships, while their survival relies, to a wide extend, on the survival of the rest as one entity, an ecosystem. This notion of *software ecosystems*, is gaining in popularity in industry and research. It appears to be the evolution of software development outside the borders of the traditional software company, also supported by the claim that software development becomes challenging without the use of consortia or vendors (Sawyer, 2000; Cusumano, 2004). Furthermore, it is arguably increasing productivity, widening the customer segments, accelerating development, allow for reuse, and fostering innovation (Bosch, 2009, 2010a; Kakola, 2010).

1.1 Problem formulation

This thesis focuses on the elements (i.e. actors, systems, social and technical relationships, and other infrastructure) evolved in the telemedicine services of Denmark, referred to as the *Telemedicine Ecosystem*, and investigates the problems inhibiting the development, implementation and use of telemedical technologies. It is our hypothesis that the establishment of a software ecosystem in the telemedicine ecosystem would address the challenges currently faced by the telemedical technologies and, thus, we investigate the conditions of establishing a software ecosystem in the telemedicine ecosystem and the impact these actions would have. Our main research question is:

> *Research Question (i):*
> *"How can a software ecosystem foster the development, implementation, and use of telemedicine services?"*

When we look at existing research on software ecosystems, we note that software ecosystems are either created by a (successful) company or product opening up the platform to external actors (Hanssen and Dybå, 2012; Kilamo et al., 2012; Bosch, 2009, 2010a) or by the establishment of an open source project (Jergensen et al., 2011; Kazman and Chen, 2010; Lungu et al., 2010a). In this project we investigate the necessary conditions of establishing a software ecosystem without having a successful product or project to support it, but by identifying the need for a software ecosystem. In order to do so, we need to support the establishment of all the essential elements of a software ecosystem. Therefore, we need to identify possible elements that form a software ecosystem to analyze them. To achieve that, we focus on the following research question:

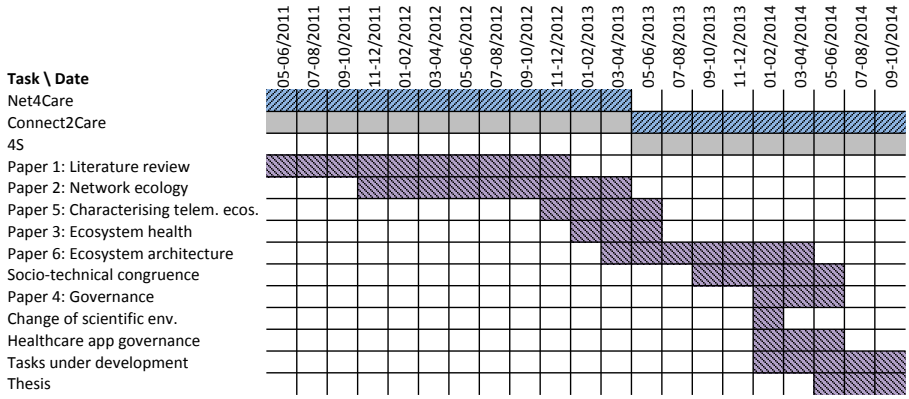


Figure 1.1: The PhD project time plan.

Research Question (ii):
"How can the concept of "software ecosystems" be defined, and how can software ecosystems be analyzed and modeled?"

In order to confirm or reject the initial hypothesis that software ecosystems can help in addressing the issues faced by the telemedicine ecosystem we need to ensure the quality of the established ecosystem. Although research question (i) is concerned with how to build the right ecosystem for the telemedicine domain, the examples of unsuccessful ecosystems, such as the Symbian ecosystem (Null, 2013) or critique on the effectiveness of the Google Play in the Android ecosystem (Mylonas et al., 2013; Orthacker et al., 2012) underline the necessity for assuring the quality of the ecosystem. To do so, we focus on the following research question:

Research Question (iii):
"How can we define and measure the qualities of a software ecosystem?"

1.2 Project time plan

This PhD project has been running from May 2011 until November 2014, including six months of leave. In total is accounted for a rough estimate of $5,000$ hours including courses for 30 ECTS, a rough estimate of 850 duty hours[2], and a change of scientific environment as a visiting researcher in Lund Technical University of Sweden. Figure 1.1 shows the time plan of the PhD project marking the major research activities[3]. This PhD project has been running as part of

[2]Duty hours are spent in lecturing, teaching assistant tasks, supervision of master students and other services to the department, group, and supervisor

[3]In this plan activities outside the major research focus has been excluded.

two research projects: Net4Care and Connect2Care[4]. Net4Care[5], as we describe in later sections, was investigating the establishment of a software platform to promote a software ecosystem of telemedicine services in Denmark and resulted in the implementation of the Net4Care platform. Connect2Care[6], among other, continued the work of Net4Care by exploiting and prototyping a connected national healthcare information technology (IT) infrastructure. These projects provided input to the 4S organization[7]. 4S is an organization established to promote the development of a software ecosystem of the telemedicine services of Denmark by managing the technological platform (Net4Care with the addition of OpenTele, explained in Paper 6) and promoting actor activity in the ecosystem.

Main outcomes of this PhD project, apart from the contribution to the two above-mentioned projects, are the included papers:

Paper 1: A systematic literature review of software ecosystems.

Manikas, K. and Hansen, K. M. (2013c). Software ecosystems – a systematic literature review. *Journal of Systems and Software*, 86(5):1294 – 1306

Paper 2: A study of software networks and use of "network ecology" theories in two open source ecosystems.

Hansen, K. M. and Manikas, K. (2013). Towards a Network Ecology of Software Ecosystems: an Analysis of two OSGi Ecosystems. In *Proceedings of the 25th International Conference on Software Engineering & Knowledge Engineering (SEKE'2013)*, pages 326–331

Paper 3: Defining and measuring the health of software ecosystems.

Manikas, K. and Hansen, K. M. (2013b). Reviewing the health of software ecosystems - a conceptual framework proposal. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 33–44

Paper 4: Studying governance of software ecosystems.

Wnuk, K., Manikas, K., Runeson, P., Lantz, M., Weijden, O., and Munir, H. (2014a). Evaluating the governance model of hardware-dependent software ecosystems – a case study of the axis ecosystem. In Lassenius, C. and Smolander, K., editors, *Software Business. Towards Continuous Value Delivery*, volume 182 of *Lecture Notes in Business Information Processing*, pages 212–226

Paper 5: Characterizing the Danish telemedicine ecosystem by studying actor and software interactions.

Manikas, K. and Hansen, K. M. (2013a). Characterizing the danish telemedicine ecosystem: Making sense of actor relationships. In *Proceedings*

[4]Start date of the projects is not shown in this figure as these projects have started before this PhD project.

[5]`http://net4care.org/`

[6]`http://www.partnerskabetunik.dk/projekter/connect2care.aspx`

[7]4S stands for "Stiftelsen for Softwarebaserede SundhedsServices" that is in English: The Foundation for Software-based Healthcare Services. `http://4s-online.dk/`

of the Fifth International Conference on Management of Emergent Digital EcoSystems, MEDES '13, pages 211–218

Paper 6: Analyzing existing ecosystems and designing new by examining the case of the Danish telemedicine ecosystem before and after the Net4Care and 4S contributions.

Christensen, H. B., Hansen, K. M., Kyng, M., and Manikas, K. (2014). Analysis and design of software ecosystem architectures – towards the 4s telemedicine ecosystem. *Information and Software Technology*, 56(11):1476 – 1492

Furthermore, in papers not included in this thesis, we have studied socio-technical congruence in software ecosystems, i.e. whether there is a correlation between the coordination of the ecosystem actors and the coordination of the software components, published in Syeed et al. (2014) and app store governance mechanisms for mission critical or healthcare apps, published in Manikas et al. (2014).

1.3 Method

The overall method of the study can be characterized as a mix-method of both qualitative and quantitative studies including literature surveys, case studies, and experiments.

The evolution of the project is to a great extent influenced by the outcomes of the systematic review on software ecosystems that is presented in Paper 1. The purpose of this review is to provide a systematic overview of the research conducted on the field. To do so we create a review protocol following the guidelines of Kitchenham and Charters (2007). Our review protocol consists of a set of research questions to explore, an extraction query, a list of bibliographic searches, and a set of inclusion and exclusion criteria. The review protocol is evaluated during the review process of the published article.

One of the observations from the literature reviews was that there were very few empirical studies of software ecosystems. Therefore, in Paper 2 we define methods for measuring and comparing software ecosystems. In this study, we apply the formalizations and metrics of "network ecology" in natural ecosystems to two software ecosystems to investigate whether it is possible to compare similar ecosystems. This is a technology-oriented quasi-experiment under the quantitative exploratory study category (Wohlin et al., 2012). This study is characterized as a quasi-experiment as the assignment of the treatments to the subjects is not random but based on the subjects themselves (Wohlin et al., 2012; Campbell et al., 1963). We define the concept of an ecosystem "neighborhood" and apply network ecology metrics to these neighborhoods. We mine the repository of Maven build automation tool, and extract the dependencies for two OSGi ecosystems: Apache Felix and Eclipse Equinox. In total, we extracted 155,127 unique artifacts and 495,020 dependencies among them from a total of 186,922 configuration files.

Continuing in the same line of measuring the quality of an ecosystem, we focused on the concept of "ecosystem health". In Paper 3 we conduct a literature review on the concept of software ecosystem health and the literature that

inspired the literature of ecosystem health and propose a conceptual framework for defining and measuring the health in a software ecosystem. To do so, we extract the software ecosystem health literature from the literature review in Paper 1 and in continuation we define the wider ecosystem health literature, i.e. the literature that inspired the software ecosystem health, applying the "snowballing technique" (Denscombe, 2010). We review in total 23 papers, 13 in the field of software ecosystems and 10 from the fields of business ecosystems, natural ecosystems, and open source software.

Another concept that influences the health of the ecosystem and the ability of the ecosystem to survive over time is the 'governance' of the ecosystem. In Paper 4 we present a case study on software ecosystem governance. We conducted an exploratory case study, following the case study process of Runeson et al. (2012), to evaluate the governance model of the Axis Camera Application Platform (ACAP) ecosystem, an ecosystem governed by Axis, a Swedish network video and surveillance company. In this study, we conducted ten exploratory interviews of practitioners knowledgeable in the ACAP ecosystem and eight interviews with external developers, developing the ACAP applications as well as discussions with the Axis employees. We compared the governance activities of the ACAP ecosystem with the governances activities of software ecosystems outlined by Jansen and Cusumano (2012, 2013).

Having obtained knowledge on how to define the health of an ecosystem (Paper 3) and how to graph ecosystem activity and identify components with important contribution to the ecosystem (Paper 2), we applied this knowledge to the Danish telemedicine ecosystem in paper 5. We characterize the actor and software interaction of the existing Danish telemedicine ecosystem. We do so by conducting a qualitative case study of the actors and the applications of the ecosystem. The purpose is to characterize the Danish ecosystem as a whole by characterizing the network of actors and applications. Our initial hypotheses are that the ecosystem under study has software systems that are primarily unrelated and that the actor structure consists of isolated components. Our study focuses on the Capital Region, one of the five healthcare regions of Denmark, responsible for one third of the country's population. Our dataset contains 28 applications, 109 actors, and 182 connections.

Paper 6 analyzes the concept of "software ecosystem architecture" as a way of modeling and analyzing existing ecosystems and designing new ones. This is done through a mix-method study: a descriptive case study on the Danish telemedicine ecosystem and an experiment of creating an orchestrator of a software ecosystem for the Danish telemedicine ecosystem. Using the theory of the software ecosystem architecture, we conduct the case study following the guidelines of Yin (2013), where we analyze the existing ecosystem of the telemedicine services in Denmark and in continuation we describe the experiment of creating 4S, an organization to serve as an orchestrator in a software ecosystem for the telemedicine services of Denmark operating on top of the Net4Care platform.

Finally, we conducted an experiment of designing, implementing, and evaluating the architecture of a platform to serve as the common technological platform and promote the establishment of a software ecosystem in the telemedicine services of Denmark. To do so, we reviewed a number of existing applications and systems (FMK (SSI, 2014a), RRS (2009), TELEKAT (2007), VAGUS, Telesår (2006), EgenJournal (2010), Sundt Hjem) and standards (HL7 (2014), CDA (Boone, 2011), PHMR (HL7, 2010), and XDS (IHE, 2013)), used the

application network in the actor-application qualitative study of Paper 5, and interviewed a SMB in the field of telemedical application development (ViewCare, 2011). To design our platform we used a set of prioritized quality attribute scenarios as output of a *quality attribute workshop* (Barbacci et al., 2003) and applying the approach of *attribute-driven design* (Bass et al., 2003). The platform is evaluated by a series of actors and applications explained in Paper 6.

1.3.1 Validity analysis

In this PhD project we have taken a series of steps to address the validity of our project. By doing so, we ensure that our methodology is solid, that we have minimized possible biases, that our results are reliable and generalizable, and our study and results are close to being repeatable, to the extent allowed from the circumstances. Below we discuss the validity aspects of our project.

Reliability

Reliability is an aspect of research validity that is concerned with addressing different kinds of bias in a study (Runeson et al., 2012; Robson, 2011). One kind of bias that is applicable to our study is the research bias in the data collection and analysis methodologies. To address this, several of the measures we have applied, we have developed and tested in other ecosystems. For example, we identify influential actors in the Danish telemedicine ecosystem in Paper 5, by applying the PageRank algorithm. This algorithm has been applied in Paper 2 where we compare it with the Keystone Index, while in Paper 5 we apply it in parallel to the Eigenvector Centrality used in related work. Furthermore, we made sure to publish each of our studies in peer reviewed journals or conferences of the field.

Construct validity

Construct validity is concerned with whether the applied measures in a study are measuring what the researchers intended (Yin, 2013; Runeson et al., 2012). This can be translated into two questions: whether the right measures are applied and whether the measures are applied right. To evaluate whether we apply the right measures, the results of several of our applied measures have been evaluated in other ecosystems, as explained in the pervious section.

To address the question on whether our measures are correct, we use a source triangulation method (Patton, 2002; Denzin, 1978) to characterize the telemedicine ecosystem in Paper 6: the quantitative analysis from Paper 5 in combination with a case study consisting of interviews and publicly available records (cf. method of Paper 6).

Internal validity

Internal validity is concerned with whether the applied measures made a difference in the study results and whether there were other not controlled factors that influenced the results of the study (Campbell et al., 1963; Runeson et al., 2012). Campbell et al. (1963) propose a number of variables to be controlled for addressing internal validity. From these variables, we examine *history* that applies to our studies.

History is described as the account of events outside the experiment occurring between two studies and having an influence to the results (Campbell et al., 1963). In our case this is applicable to the comparison of the two telemedicine ecosystems. We initially characterize the Danish telemedicine ecosystem at the time of the project initiation (i.e., before the Net4Care and 4S experiments) and in continuation we analyze how ecosystem can be after our experiments. There is a probability for threats to the internal validity of our study, if between the two characterizations of the ecosystems external factors would affect the ecosystem. An example would be a change in legislation or ecosystem governance that would attract more actors to the ecosystem or the opposite. This threat is minimized in our study by validating our platform with actors both internal to the project, thus unaffected from possible changes to the telemedicine ecosystem and external actors, already part of the ecosystem.

External validity

External validity or generalizability is the aspect of research validity that is focusing on the extent the results of a study can be applicable to other cases (Robson, 2011; Yin, 2013). We address possible issues to external validity by following an approach where we build the elements of our study's method initially as contributions to the field of software ecosystem and then apply it to our domain. An example is the definition of ecosystem health consisted, among others, from the social networks of actor and software interaction. This definition was first published as a conceptual framework for software ecosystem health (Paper 3) and then applied parts of it into Paper 5. Moreover, the main results of our study in the domain of telemedicine were generalized and encapsulated in the concept of software ecosystem architecture and published in a special issue on software ecosystems in Paper 6. Furthermore, we have tried to provide as much information we can on the particularities of our cases as means of assisting the generalization of our methods and results from external researchers. Finally, we have not, to this point, examined generalizability of our study to similar domains, such as the telemedicine services of other countries.

1.4 Summary

Concluding the introduction, the demographic changes in Denmark and the organizational and financial changes of Danish healthcare are pointing towards the establishment of telemedicine as means of patient support, diagnosis, and treatment. However, telemedical solutions are faced with a number of challenges that inhibit their adoption, development, and implementation. We investigate means of addressing these challenges following the approach of software ecosystems, roughly explained as the software development and distribution by a set of actors dependent on each other and the ecosystem. In order to do so we conducted both quantitative and qualitative studies including literature surveys, case studies, and experiments.

This work concludes on contributions initially to the theoretical domain where we define and explain software ecosystems, provide means of analyzing existing and designing new ecosystems, and define and measure their qualities. Based on these contributions, we design an ecosystem to be established

in the telemedicine ecosystem characterized by a common platform, increased software interaction, stronger social networks of actors, robust business structures, and orchestration and governance to promote the health of the ecosystem. Furthermore, we take the initial steps in materializing this design with the design, development, and implementation of Net4Care, a platform to serve as the common platform in the software ecosystem. Moreover we provide input to 4S organization, an organization to serve as an orchestrator in the ecosystem managing the platform, supporting actor and software interactions, and promoting the ecosystem health.

The rest of this part is organized as following. Chapter 2 contains the state of the art in the field of software ecosystems and updates the literature review of Paper 1 to reveal an increased research activity in the field and early signs of the field maturing in the past two years. Chapter 3 characterizes the Danish telemedicine ecosystem by initially analyzing the general healthcare in Denmark and then analyzing the telemedicine ecosystem in two snapshots: first the current, before the Net4Care and 4S influence, and second towards a software ecosystem analyzing the influence of Net4Care and 4S. Finally, Chapter 4 concludes this part by summarizing the contributions and explaining how they address the research questions, discussing implications and threats to validity, and elaborating on future work. Part II of this thesis contains the published articles in the sequence they appear in this chapter.

Chapter 2

Theory: Software ecosystems

In this chapter we introduce and explain in more detail the field of software ecosystems. We do so by providing a definition and describing the main elements of a software ecosystem. Moreover, we apply the protocol of our literature review, explained on Paper 1, to update the literature of software ecosystems. Comparing our results with the results of Paper 1, we note that within two years the literature in the field has expanded significantly, while we note indications of field maturity.

The way software is produced and distributed has been radically altered the past years. Initially, software was developed "in-house" by a team of highly domain-specialized engineers and either came embedded in the hardware or distributed as a product to be installed. Slowly, software started becoming more compartmentalized and different companies would build different expertise. Thus, one could develop software by either hiring an expert to build parts of it (outsourcing), e.g. the user interface, or buying and incorporating a ready built product (commercial-off-the-shelf), e.g. a database. In this case, the distribution is still through the traditional product model. With the advent of the cloud computing, one can now buy services (Software as a Service) online. In this case the development is done in one of the ways mentioned above, but the distribution is done by installing the software on a cloud server. A software ecosystem challenges the ways software is developed and distributed. Software is developed by different actors (such as companies or developers), while the owner of the platform is not necessarily the owner of the developed products or the one who pays for the development. Moreover, software might be distributed as a product or a service, but it cannot stand alone without the ecosystem platform.

Software ecosystems have been gaining popularity in recent years changing the software development and distribution. Known examples are smart phone platforms and open source software. The advance of mobile platforms like the platforms supporting Apple Appstore and Google Play made software products highly commercialized, approachable, and easy to develop. The open source software (OSS) projects like the Eclipse and Firefox are examples of software development and management that is collaborative and outside the borders of the traditional software company.

Figure 2.1: The software ecosystem keyword word cloud.

A software ecosystem can be defined as "the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services". Moreover, "each actor is motivated by a set of interests or business models and connected to the rest of the actors and the ecosystem as a whole with symbiotic relationships, while, the technological platform is structured in a way that allows the involvement and contribution of the different actors" (Manikas and Hansen, 2013c).

In other words, a software ecosystem can be identified as any software development and distribution, including the socio-technical environment around it, that is characterized by the following components: (i) a platform that allows development, (ii) a set of actors that are connected to the ecosystem with symbiotic relationships, and (iii) a set of motives or business models serving the actors. Software ecosystems can be very different, varying in any or all of the above components. For example an ecosystem can have a platform that is a software product itself, like the case of Microsoft Excel (Bosch, 2009) or the Apache web server (Yu, 2011), that is bound to specific hardware, like the hardware-dependent ecosystem of Axis (Wnuk et al., 2014a) or the embedded open software ecosystems (Eklund and Bosch, 2014), or a framework like the case of the Open Design Alliance (Angeren et al., 2011b; Jansen et al., 2012) or OSGi (Jansen and Cusumano, 2012). Similarly, examples of the actor relationship variability can be an ecosystem that is open to actor participation, like several OSS projects or the Google Play ecosystem, or closed where new actors are selected according to a set of criteria, like the Microsoft Partner Network. From the business model perspective, an ecosystem could be providing opportunities to the actors for direct benefits, like the Apple Appstore for app developers, or provide other than monetary motivation/satisfaction like several OSS project participants. Naturally, the different ecosystem variability aspects mentioned here are only examples and by no mean provide an exhaustive list.

In order to provide a better overview of the literature, in the following section we conduct a review of the literature on software ecosystems up to date. We do so by expanding on the review of Paper 1 using the same review protocol.

2.1 Literature overview

The concept of a 'software ecosystem' first appeared in the book by Messerschmitt and Szyperski (2003). Since then the field has been growing to include two dedicated workshops (IWSECO, 2014; WEA, 2014), a number of conferences, and two special journal issues. Paper 1, gives an overview of the field of software ecosystems from the first paper in 2007 until June, 2012 reviewing a total of 90 papers out of a gross of 420. For the needs of this thesis, we expanded the literature using the same systematic literature review protocol, to include published literature up to June 2, 2014. In total, the software ecosystem literature counts 199 papers (i.e. 109 papers from June 2012 to June 2014). In this overview we only reviewed the abstract and keywords of the resulted papers. The purpose of the current review is to provide a condensed and up-to-date state of the art in the research of software ecosystems. In Figure 2.1, we show the word cloud extracted from the keywords fields of the software ecosystem literature to provide a quick overview on what are the most common topics in the literature. For reasons of readability we have removed the words "software"

Table 2.1: Papers published per year

Year	Papers	Count
2007	[31, 46, 57]	3
2008	[50, 53, 54]	3
2009	[9, 10, 11, 20, 25, 42, 51, 52, 56, 58]	10
2010	[1, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 27, 28, 29, 30, 33, 34, 35, 36, 37, 38, 39, 41, 43, 45, 47, 48, 49, 55, 59]	32
2011	[2, 3, 4, 5, 6, 7, 8, 26, 32, 40, 44, 87, 89, 88, 73, 72, 74, 68, 71, 69, 64, 61, 65, 70, 79, 66, 60, 67, 83, 85, 90, 84]	32
2012	[63, 80, 86, 82, 76, 77, 75, 62, 78, 81, 107, 108, 115, 116, 117, 119, 124, 128, 129, 130, 133, 134, 135, 142, 147, 149, 181, 158, 162, 163, 170, 173]	32
2013	[91, 92, 93, 95, 96, 97, 98, 99, 100, 101, 103, 104, 105, 106, 109, 110, 112, 118, 120, 123, 125, 126, 127, 131, 137, 138, 139, 140, 141, 144, 148, 151, 153, 154, 155, 194, 195, 196, 197, 198, 199, 178, 187, 156, 157, 160, 165, 167, 169]	49
2014	[94, 102, 111, 113, 114, 121, 122, 132, 136, 143, 145, 146, 150, 152, 159, 161, 164, 166, 168, 171, 172, 174, 175, 176, 177, 179, 180, 182, 183, 184, 185, 186]	32

and "ecosystem(s)".

2.1.1 Publications per year

An indication of the focus on a field is to measure the number of publications per year. Table 2.1, shows the publication according to the year of publication. We notice that the number of publications has been constantly increasing from 2007. Naturally, the year 2014 is not taken into consideration as the full number of publications for that year is not yet available. The above indicate that the focus on the field is increasing.

2.1.2 Research result types

Following the same method with Paper 1, we have classified each of the papers in the literature according to their research result using the Shaw (2003) characterization of software engineering research results. Table 2.2 explains the characterizations.

The classification of software ecosystem results is shown in Table 2.3. If we compare the results with the ones from Paper 1, we note that there is an increase in the percentage of the groups 'report' (from 44% to 49%), 'empirical model' (from 8% to 15%), while there is a decrease in 'tools or notation' (from 15% to 11%), 'qualitative or descriptive models' (from 11% to 6%), 'analytic models' (from 5% to 3%), and 'specific solution' (from 4% to 2%).

We can see that the majority of the literature results is 'report'. This is mainly because many of the papers are based on single studies that, although interesting, are hard to generalize. An example is Wnuk et al. (2014b), where they study the actor participation model of ACAP, the app ecosystem around the monitoring and surveillance hardware producer, Axis. Although interesting,

Table 2.2: Types of research results according to Shaw (2003).

Type of Result	**Examples**
Procedure or technique	New or better way to do some task, such as design, implementation, maintenance, measurement, evaluation, selection from alternatives; includes techniques for implementation, representation, management, and analysis; a technique should be operational—not advice or guidelines, but a procedure
Qualitative or descriptive model	Structure or taxonomy for a problem area; architectural style, framework, or design pattern; non-formal domain analysis, well-grounded checklists, well-argued informal generalizations, guidance for integrating other results, well-organized interesting observations
Empirical model	Empirical predictive model based on observed data
Analytic model	Structural model that permits formal analysis or automatic manipulation
Tool or notation	Implemented tool that embodies a technique; formal language to support a technique or model (should have a calculus, semantics, or other basis for computing or doing inference)
Specific solution, prototype, answer, or judgment	Solution to application problem that shows application of SE principles – may be design, prototype, or full implementation; careful analysis of a system or its development, result of a specific analysis, evaluation, or comparison
Report	Interesting observations, rules of thumb, but not sufficiently general or systematic to rise to the level of a descriptive model.

their results are too specific to the ecosystem under study and hard to generalize to other ecosystems. Moreover, the categories can be, to some extent, specific for software engineering. Thus, papers not reporting software engineering results might end up in the 'report' group. An example is the group 'specific solution', that is described as a "solution to application problem that shows application of SE principles" (Shaw, 2003).

A group that is not described as software engineering specific, is the 'empirical model'. Examining the increase of this group implies that more existing ecosystems have been studied in such a way that the results can be applied to other ecosystems. This is an early indication that the field has been maturing these two years.

2.1.3 Software ecosystem classification

In Paper 1, when examining what is the role of architecture in the software ecosystem research, we propose the concept of "software ecosystem architecture", by expanding the definition of software architecture and identifying three main categories to analyze the literature: software engineering, business and management, and software ecosystem relations. Paper 6 expands on this definition and models the architecture of a software ecosystem using three structures, somewhat different from the categories above: software structure, business structure, and organizational structure, where the organizational structure incorporates, among other, the actor relationships of the ecosystem. We believe that the latter is more representative to reason for a software ecosystem, however using the

Table 2.3: The papers grouped according to the result types.

Result	**Papers**	**%**
Procedure or Technique	[40, 41, 5, 11, 30, 53, 16, 13, 28, 18, 14, 17, 101, 107, 109, 111, 116, 119, 129, 131, 133, 134, 137, 149, 174, 177, 192]	13
Qualitative or Descriptive Model	[38, 39, 3, 4, 7, 21, 37, 29, 34, 74, 164, 183]	6
Empirical Model	[45, 50, 44, 57, 55, 27, 89, 94, 98, 102, 104, 112, 115, 121, 127, 132, 140, 146, 147, 151, 157, 158, 169, 170, 179, 180, 182, 184, 193, 199]	15
Analytic Model	[1, 42, 63, 72, 90, 172, 187]	3
Tool or notation	[9, 48, 58, 54, 15, 22, 47, 25, 80, 68, 69, 62, 79, 81, 100, 106, 108, 113, 135, 139, 153, 155]	11
Specific solution, prototype, answer, or judgment	[49, 23, 33, 88]	2
Report	[19, 43, 2, 6, 10, 8, 59, 52, 56, 51, 36, 46, 31, 20, 35, 12, 26, 32, 24, 14, 29, 33, 17, 87, 73, 86, 82, 76, 77, 71, 75, 64, 61, 65, 70, 78, 66, 67, 83, 85, 84, 91, 92, 93, 95, 96, 97, 99, 103, 105, 110, 114, 117, 118, 120, 122, 123, 124, 125, 126, 128, 130, 136, 138, 141, 142, 143, 144, 145, 148, 150, 152, 154, 156, 159, 160, 161, 162, 163, 165, 166, 167, 168, 171, 173, 175, 176, 178, 181, 185, 186, 188, 189, 190, 191, 194, 195, 196, 197, 198]	50

Table 2.4: The papers according to the software ecosystem architecture groups.

Architecture Group	**Papers**	**Count**
SE	[46, 15, 22, 47, 35, 29, 17, 57, 87, 63, 68, 69, 70, 79, 66, 81, 94, 95, 97, 99, 100, 101, 105, 106, 107, 108, 109, 110, 111, 112, 113, 115, 116, 117, 118, 119, 120, 121, 122, 124, 125, 131, 133, 135, 136, 137, 138, 139, 140, 143, 144, 145, 146, 147, 148, 149, 152, 153, 155, 159, 160, 162, 163, 166, 168, 169, 170, 171, 173, 174, 178, 179, 180, 181, 182, 183, 184, 187, 190, 194]	80
Business and management	[20, 23, 37, 44, 14, 33, 12, 25, 27, 87, 88, 86, 76, 77, 71, 75, 62, 66, 60, 67, 83, 90, 91, 93, 112, 114, 116, 123, 124, 126, 127, 128, 130, 131, 132, 141, 142, 145, 151, 154, 156, 158, 160, 161, 164, 165, 166, 167, 168, 171, 174, 175, 177, 178, 179, 180, 183, 184, 185, 186, 188, 189, 191, 192, 193, 196, 197]	67
Relationships	[13, 31, 21, 33, 17, 87, 73, 86, 82, 72, 76, 61, 65, 85, 84, 91, 92, 93, 96, 98, 102, 103, 104, 121, 122, 125, 129, 133, 134, 141, 150, 151, 157, 163, 172, 176, 178, 183, 184, 185, 188, 189, 190, 194, 195, 198, 199]	47

Table 2.5: Top 25 paper keywords according to group.

Software Engineering		**Business and Management**		**Relationships**	
Keyword	**Count**	**Keyword**	**Count**	**Keyword**	**Count**
architecture	42	development	25	development	15
systems	29	source	20	social	12
data	20	open	19	systems	12
computing	20	modeling	17	engineering	10
evolution	18	engineering	16	open	10
product	16	mobile	15	industry	9
model	16	computing	14	network	9
development	15	systems	13	analysis	8
lines	14	management	13	based	8
platform	13	architecture	11	data	8
mobile	13	governance	8	architecture	7
engineering	12	communities	8	modeling	7
based	11	product	8	source	7
maintenance	10	application	7	collaboration	6
management	10	network	7	communities	6
mining	10	project	7	management	6
variability	10	business	7	classification	5
packages	9	health	7	environment	5
processing	9	data	6	reuse	5
industry	8	domain	6	value	5
analysis	8	oss	6	visualization	5
code	8	value	6	aspects	4
quality	8	industry	6	computing	4
repositories	8	case	5	global	4
application	8	disease	5	knowledge	4

same classification as Paper 1 gives a consistency in the analysis of the literature. Thus, below we expand on the literature according to the three categories of Paper 1. The papers used to describe each group might not necessarily represent the papers' main focus.

Software engineering

Software plays an important role in a software ecosystem. The product of software ecosystems is software, while ecosystems are evolved around a common software platform. The software engineering category includes literature covering different software engineering aspects of software ecosystems. Table 2.5 lists the 25 most common keywords taken from the 'keywords' field of each paper for the three groups. From the list of keywords we have removed the two most common words "software" and "ecosystem(s)" in all three groups. The most

common keywords for the group 'software engineering' after that is "architecture", along with the similar of "architectural" and "architectures". The words appear in the papers' keyword fields as "open architecture" (Pelliccione, 2014), "parallel architecture" (Bortolotti et al., 2013), "service oriented architecture" (Schmerl et al., 2011; Kajan et al., 2011), and "software architecture" (Musil et al., 2013; Gutierrez and Robbes, 2013; Christensen and Hansen, 2012; Santos and Werner, 2012a; Eklund and Bosch, 2014; Cataldo and Herbsleb, 2010; Kazman et al., 2012; Scacchi and Alspaugh, 2012)[1]. The software architecture of a software ecosystem should support the nature of the ecosystem (i.e. be adapted to the needs of the specific ecosystem), follow the ecosystem management, business rules and restrictions and allow the integration and existence of multiple functionality in a secure and reliable manner. A modular and flexible architecture would allow integration and interoperability of the developed software (Viljainen and Kauppinen, 2011; Bosch, 2009). An aspect of software engineering in software ecosystems that is related to software architecture, is the design and use of interfaces and application programmable interfaces (APIs) [2]. Interfaces allow external development on an ecosystem platform. The stability and translucency of the platform interfaces are essential for the component integration and interaction (Cataldo and Herbsleb, 2010; Bosch, 2010a). Changes to existing interfaces or components might create inconsistencies to dependent components (Robbes and Lungu, 2011; Lungu et al., 2010b,a). Process-centric approaches are not effective in managing large-scale software, instead system architecture should be used as a coordination mechanism (Bosch and Bosch-Sijtsema, 2010a). Constantly evolving software requires the adaptation of the software development processes. Development should be integration-centric, independent deployment and releases should be organized in a release grouping and release train fashion (Bosch and Bosch-Sijtsema, 2010b; Bosch, 2010a). Architectural design and analysis techniques are based on a set of principles as identifying business goals, describing architectural significant requirement, tactics and architectural evaluation. These principles are used in defining the software architecture of a software ecosystem (Kazman et al., 2012).

Evolution in software ecosystems is an aspect that draws the attention of a number of papers[3]. Evolution is studied as the evolution of the software ecosystem by means of requirement negotiation (Valenca, 2013) or by means of visualizing the evolution of code in an OSS ecosystem (Pérez et al., 2012). Moreover, software evolution, i.e. how software changes over time, is also of focus, mainly in OSS projects. Software evolution is studied by directly examining the evolving code, studying how the interfaces and APIs evolve in an ecosystem, or examining the evolution of software dependencies.

The intersection of software ecosystems and software product lines is the fo-

[1]For reasons of text readability, only a few papers with keywords "software achitectur-(e)(es)(al)" are listed.

[2]Keywords of McDonnell et al. (2013); Linares-Vásquez et al. (2014); Cataldo and Herbsleb (2010); German et al. (2013); Peniak et al. (2013); Bavota et al. (2013).

[3]Keywords of Pelliccione (2014); Seidl and Aßmann (2013); Seidl et al. (2013); Gutierrez and Robbes (2013); Lopez (2013); German et al. (2013); Bavota et al. (2013); Mens et al. (2014a); Schwarz et al. (2012); McDonnell et al. (2013); Mens et al. (2014b); Goeminne and Mens (2010); Lungu and Lanza (2010); Scacchi (2010b); Brummermann et al. (2011); Lungu et al. (2010a); Yu et al. (2007); Brummermann et al. (2012); Yu (2011); Scacchi and Alspaugh (2012); Pérez et al. (2012); Neu et al. (2011).

cus of yet another set of research papers[4]. A software product line can evolve to become a software ecosystem with the product line being the common platform of the ecosystem. In this intersection, variability in software ecosystems and product lines and the variability modeling and management is the focus of a part of the literature[5].

Another part of the software ecosystem literature is focusing on issues that are appearing mainly in OSS. One of the methods used to conduct empirical studies in this field, is mining software repositories[6]. OSS repositories can be a valuable source of information as they contain or it is possible to extract information about the technical and social perspectives of the project, such the sources, dependencies, bugs and changes, developer communications, or mailing lists. One feature of OSS repositories is the option to fork or copy a whole project and start a different branch. Keivanloo et al. (2012) and Schwarz et al. (2012) are focusing on the code cloning along repositories. A similar perspective to cloning, the reuse of software is also studied in the literature [7]. Software reuse is a field that is not exclusive for OSS, however, the biggest amount of literature focuses on OSS.

Integration is occupying another part of the literature. Integration appears as integration-centric approaches (Eklund and Bosch, 2012), strategies for (software) integration (Bhowmik et al., 2014), software integration (Bosch and Bosch-Sijtsema, 2010b; Capuruço and Capretz, 2010), integration of knowledge resources (Schugerl et al., 2009), and integration of platforms (Viljainen and Kauppinen, 2011).

Quality in software ecosystems is also on focus (Hmood et al., 2012; Hansen and Zhang, 2014; Schugerl et al., 2009; Stefanuto et al., 2011; Jansen, 2013). The quality of the produced software in an ecosystem and how to measure it and assure high levels is the main focus here (Hmood et al., 2012; Schugerl et al., 2009; Kajan et al., 2011).

Finally, the field of software engineering, having to do with requirement, requirement engineering is also represented in the ecosystem literature (Valenca, 2013; Fricker, 2009; Schugerl et al., 2009).

Business and management

Another group of the software ecosystem literature is the business and management of software ecosystems. A software ecosystem needs to be organized and managed in some way, whether it is driven by for-profit organization or a non-profit community. Moreover, an essential element that makes ecosys-

[4]Keywords of Lettner et al. (2013); Seidl et al. (2013); Keunecke et al. (2013); Berger (2012); Kästner et al. (2012); Santos and Werner (2012a); Schultis et al. (2013); Schmid (2013); Berger et al. (2014); Brummermann et al. (2012)

[5]Keywords of Kästner et al. (2012); Seidl and Aßmann (2013); Seidl et al. (2013); Keunecke et al. (2013); Berger (2012); Kästner et al. (2012); Schmid (2013); Berger et al. (2014); Brummermann et al. (2011, 2012).

[6]Keywords of Teixeira and Lin (2014); Goeminne et al. (2013); Keivanloo (2012); Mitropoulos et al. (2014); Lopez (2013); Linares-Vásquez et al. (2014); Tymchuk et al. (2014); Hoving et al. (2013); Bavota et al. (2013); Keivanloo et al. (2012); Goeminne (2014); Berger et al. (2014); Santana and Werner (2013); Syed and Jansen (2013); Goeminne and Mens (2010); Lungu et al. (2009); Hindle et al. (2010); Robbes and Lungu (2011).

[7]Keywords of Santos and Werner (2012b); Kakola (2010); Santos and Werner (2012a); Costa et al. (2013); Santos (2014); Santos et al. (2013); Albert et al. (2013); Bhowmik et al. (2014); Santos and Werner (2011a).

tems successful is the business or motivation that provides the incentive for the actors to contribute to the ecosystem. Table 2.4 shows the papers that are focusing on this group, while Table 2.5 lists the most common keywords of this category, again excluding the words "software" and "ecosystem(s)". From that table we note that the literature has more distributed keywords, and thus more subjects of study, comparing to the software engineering group. The most common keywords is "develop-(ment)(er)". Looking at the literature, we find that apart from classifying the papers under software development (Santos et al., 2012; Bhowmik et al., 2014; Schugerl et al., 2009), this keyword is also used in a number of papers addressing issues related to development involvement and motivation (Fagerholm and Munch, 2012; Ververs et al., 2011). Looking at the keywords for "model-()(s)(ing)(ling)" we note that this keyword is used in several different studies: Business models and issues related to that in the software ecosystem context (Frantz and Corchuelo, 2012; Andresen et al., 2013; Handoyo, 2013) and how to model software ecosystems[8].

Not surprisingly, the keyword "management" also appears in the top keywords. Management appears as management of the software creation process (software/project/development management)[9] or as more general ecosystem aspects (partner/customer relationship/knowledge/software ecosystem asset/co-creation management)[10].

The keyword "network" appears as analyzing the software supply network of software ecosystems (Handoyo et al., 2013a; Boucharas et al., 2009) but also the social networks of actors (Santos et al., 2012; Scholten et al., 2012). These are keywords that also relate to the last group of software ecosystem: Relations, therefore underlining the overlap the two groups have.

The keyword "open" relates more to the applied domain of a study, like open source, however, there are a number of studies that relate to the "openness" of an ecosystem, i.e. how open the ecosystem for external actors (Anvaari and Jansen, 2010; Jansen et al., 2012).

"Health" is a keyword that mainly refers to the concept of health of an ecosystem, a central notion in defining the well functioning of an ecosystem (Jansen, 2014; Manikas and Hansen, 2013b; Lingen et al., 2013). In order to ensure that a software ecosystem is functioning well, specific measurements need to be introduced that would provide an overview of the state of the ecosystem while at the same time raise attention for actions and allow comparison of ecosystems. This concept has been introduced by Iansiti et al. as a way to measure the performance of a business ecosystem. In more detail they measure the "*extent to which an ecosystem as a whole is durably growing opportunities for its members and those who depend on it*" (Iansiti and Levien, 2004a) and inspired from biological ecosystems define the health of a (business) ecosystem as an analogy to robustness, productivity and niche creation (Iansiti and Levien, 2004b; Iansiti and Richards, 2006; Iansiti and Levien, 2004c,a). These studies, although excluded from the collected literature, are referenced by the literature elaborating on software ecosystem health[11]. An additional study on the health of

[8]Keywords of Handoyo et al. (2013a); Costa et al. (2013); Santos et al. (2012); Mens et al. (2014a); Handoyo (2013); Bosch and Bosch-Sijtsema (2014); Boucharas et al. (2009).

[9]Appearing in Bavota et al. (2013); Fagerholm and Munch (2012); Albert et al. (2013).

[10]Keywords of Nöhren et al. (2014); Schütz et al. (2013); Ingen et al. (2011); Mizushima and Ikawa (2011); Albert et al. (2013).

[11](Ingen et al., 2011; Angeren et al., 2011a; Berk et al., 2010; Kilamo et al., 2012; Jansen

business ecosystems that is referenced by several papers of the literature[12] that are elaborating on software ecosystem health, is the paper of den Hartigh et al. (2006). Based on the Iasiti et al studies mentioned above, den Hartigh et al. (2006) apply health measurement to Dutch IT business ecosystems. In the software ecosystem field, Berk et al. (2010) base their work on business ecosystem health to create a strategy assessment model. Manikas and Hansen (2013b) review the literature of software ecosystem health and the literature for health in other types of ecosystems and propose a conceptual framework for defining and measuring the health in software ecosystems. Jansen (2014) proposes the "Open Source Ecosystem Health Operationalization" a framework for operationalizing the health of OSS ecosystems.

Finally, the ecosystem "governance" is a representative focus of this category[13]. The proper governance of a software ecosystem will make proper use of the ecosystem resources, enhance productivity in the ecosystem, support robustness, and promote the ecosystem health. Baars and Jansen (2012) propose a framework to analyze the governance of software ecosystems for individual companies. The framework is consisted of five categories: explicitness of the ecosystem, explicitness of the governance, responsibility, measurement and knowledge sharing. Jansen et al. (2012) propose a model for measuring the degree of openness of a software producing organization. The model lists five areas where the organization can open up examined under three levels: strategic, tactical and operational. One of these areas is governance, where different options for opening up the governance are discussed for the three levels. Jansen (2013), based on the two previous papers extract four classification factors for software ecosystems: underpinning technology, (type of) coordinators, extension market and accessibility (of the ecosystem). In continuation, they propose a governance model for the prevention and improvement of the ecosystem health. In this model they distinguish between a software (service) platform and a standard as the types of underpinning technology and a community or private entity for each technology. For the two separations (technologies and community/private entity), they propose actions that address each of the Iansiti and Levien health measures: niche creation, productivity, robustness Iansiti and Levien (2004c). Wnuk et al. (2014a) apply the above governance model to the Axis Application Development Partner (ADP) ecosystem, a hardware-centric ecosystem with product distribution particularities, and conclude that although the above model is useful in characterizing the governance of ADP, the model needs to be evolved to capture particular aspects of ecosystems like ADP.

Software ecosystem relationships

An important part of software ecosystems is the involved actors of the ecosystem and the relationships among them and with the ecosystem. This "social" perspective of ecosystems affects and is affected to a big extend by the other two groups. The ecosystem management can, for example, define how easy it is for new actors to be involved in the ecosystem, the business models and mo-

et al., 2012, 2009a; Viljainen and Kauppinen, 2011; Mizushima and Ikawa, 2011; McGregor, 2010; Santos and Werner, 2011b; Boucharas et al., 2009)

[12](Angeren et al., 2011a; Berk et al., 2010; Kilamo et al., 2012; Jansen et al., 2012, 2009a)

[13]Keywords of Albert et al. (2013); Baars and Jansen (2012); Wnuk et al. (2014a); Jansen et al. (2012); Jansen and Cusumano (2012).

tivation can attract new actors, while the way software is produced and the architecture both of the common platform but also the products can influence the relationships of the actors and should reflect the ecosystem management. It is no surprise that the most common keywords for this group, shown in Table 2.5, include words like "social", "network", "collaboration", and "communities". The keyword "develop-(ment)(er)" is also one of the top keywords like in the other two groups, but in this group, apart from the traditional software development[14], there is more social perspectives like distributed software development (Teixeira and Lin, 2014) and global software development Santos and Werner (2012a) and management or practices of software development (Scacchi, 2007a; Goeminne, 2014).

The social focus on the literature appears as social networks or social aspects/perspectives (of software ecosystems)[15] and it relates mainly to studying the actor to actor or actor to ecosystem dependencies and interaction. Examining the relationships in an ecosystem as a network is also an approach that is followed by papers in this category[16] or looking at a more specific network the software supply network (Angeren et al., 2011a; Handoyo et al., 2013a). The collaboration in an ecosystem context is a relevant focus for this group[17], many times relating to software development. Another aspect of relationships in software ecosystems is the community[18], many times relating to an OSS project.

The following list gives an overview of the most common actors encountered in the literature.

Orchestrator , "keystone (player, organization)", "hub", "shaper", "management (unit)", or "platform owner" is a company, department of a company, actor or set of actors, community or independent entity that is responsible for the well-functioning of the software ecosystem. This unit is typically managing the ecosystem by running the platform, creating and applying rules, processes, business procedures, setting and monitoring quality standards and/or orchestrating the ecosystem's actor relationships .

Niche player , "influencer", or "component developer/builder/team" is the actor that contributes to the ecosystem by typically developing or adding components to the platform, producing functionality that customers require. This actor is part of the ecosystem and complements the work of the keystone by providing value to the ecosystem. Depending on the management model of the ecosystem the niche players might influence the decision making in the management of the software ecosystem.

External actor , "external developer (team)", "third party developers or community", "external parties", "external partner", "external entities", "participant", or "external adopter" is the actor (company, person, entity) that

[14]Keywords of Howison and Herbsleb (2013); Santos and Werner (2012b); Spauwen and Jansen (2013); Scacchi (2007a).

[15]Keywords of Cardoso et al. (2013); Santos and Werner (2012a,b); Seichter et al. (2010); Goeminne (2014).

[16]Keywords of Cardoso et al. (2013); Santos and Werner (2012a,b); Angeren et al. (2014); Seichter et al. (2010); Manikas and Hansen (2013a).

[17]Keywords of Howison and Herbsleb (2013); Tymchuk et al. (2014); Tchoua et al. (2013); Cataldo and Herbsleb (2010).

[18]Keywords of German et al. (2013); Santos and Werner (2012a,b); Tchoua et al. (2013); Goeminne (2014); Vasilescu et al. (2014).

makes use of the possibilities the ecosystem provides and thus providing indirect value to the ecosystem. This actor is external to the ecosystem management and usually has an activity limited to the actor's interest. Depending on the nature of the ecosystem, the external actor might be developing on top of or parallel to the platform, identify bugs, promote the ecosystem and its products or propose improvements. This type of actor includes the role of the participant or follower in FOSS ecosystems. An actor that is member of the ecosystem with either participation of limited responsibility or simply observing the evolution of the software ecosystem from the inside.

Vendor , "independent software vendor (ISV)", "reseller" or "value-added reseller (VAR)" is mainly the company or business unit that makes profit from selling the products of the ecosystem to customers, end-users or other vendors/VARs. The products might be complete integrations, components, selling or leasing of licenses or support agreements. A vendor that is modifying a product by e.g. adding functionality or combining different components together is called VAR.

Customer or "end user" is the person, company, entity that either purchases or obtains a complete or partial product of the ecosystem or a niche player either directly from the ecosystem/niche player or through a vendor/VAR.

An interesting perspective of software ecosystem relationships is the actor participation model that ecosystems follow. Different ecosystems apply different models for allowing actors to contribute to the ecosystem. These models are many times related to the nature of the platform and to what extent it allows/supports different kinds of collaboration, but mostly to the business model behind the ecosystem. Molder et al. (2011) propose the evaluation of how open or closed a platform and suggest a model for assessing it. Jansen et al. (2012) also study how open an ecosystem is to new actors by making a separation between the supply and demand of an ecosystem. They claim that a software ecosystem can choose to open either of them or both. They explain that the benefits of opening up the ecosystem are often not clear and a post-evaluation of whether the ecosystem was ready for the changes will be reflected in the ecosystem health after the changes have been applied.

Kabbedijk and Jansen (2011) mine the developer activity of the free or open source software (FOSS) Ruby repository, identify three developer roles and conclude with proposals for improving the health of the ecosystem. In their paper, Angeren et al. (2011a) analyze the choice of suppliers and supplier-vendor relationships the orchestrators make and identify four supplier strategies. Yu et al. (2007) evaluate the collaboration of different FOSS projects in terms of symbiotic relationships.

From the perspective of FOSS, there are a number of papers studying the FOSS participants in terms of motivation and collaboration of developers Scacchi (2010a); Mens and Goeminne (2011), what affects developer activity in a project Ververs et al. (2011); Ingen et al. (2011); Krishna and Srinivasa (2011), the organizational structure of participants Jergensen et al. (2011), and tools for extracting information or visualizing participant activity Lungu et al. (2010a, 2009); Goeminne and Mens (2010); Neu et al. (2011).

2.2 Summary

In this chapter we introduce and explain the notion of software ecosystems. We define software ecosystems using the definition proposed in Paper 1, which takes into consideration the existing definitions and different groupings of software ecosystems. Moreover, we conduct a literature overview expanding on the literature review of Paper 1 and demonstrate that the field has increased significantly in the past two years with 109 new research papers giving a total of 199. Furthermore we identify an early indication of the field becoming more mature with an increase of 'empirical models' within the last two years. Finally, we analyze the literature of the three main literature groups: software engineering, business and management, and relationships.

Chapter 3

Praxis: Telemedicine ecosystem

In this chapter, we define and analyze the applied domain of this project: the Danish telemedicine ecosystem. Telemedicine services in Denmark are part of the country's healthcare services and thus, there is a big overlap in terms of organization, financing, and possibly IT systems. Therefore, in order to define and analyze the telemedicine ecosystem, we first analyze Danish healthcare. In continuation, we analyze the telemedicine ecosystem in two ways: firstly, we explain the "current" telemedicine ecosystem, the ecosystem before the Net4Care and 4S influence (and thus, also this PhD project's influence), identifying that it is not a software ecosystem (Section 3.2.1) and secondly how the ecosystem has changed during the project and towards becoming a software ecosystem (Section 3.2.1).

To analyze the telemedicine ecosystem, we use the concept of *software ecosystem architecture*, proposed in Papers 1 and 6. A software ecosystem architecture is defined as "the set of structures needed to reason about the software ecosystem, which comprise actor and software elements, relations among them, and their properties" (Christensen et al., 2014). According to this approach, a software ecosystem may be analyzed through three structures: the organizational, business, and software structure.

The organizational structure contains elements related to the organization and governance of the ecosystem or the elements forming the ecosystem (e.g. actor and software elements). Part of the organizational structure can be the network of actors (such as developers, orchestrators, or distributors) or software components, the borders of the ecosystem, the social networks of actors and software[1], or the way the ecosystem is governed.

The business structure is related to the creation, delivery, and capture of value in the ecosystem. 'Value', refers to the benefit an actor gets from the ecosystem, such as monetary revenues or recognition. The business structure can be described through the business models serving the actors in the ecosystem, actor motivation and incentives, or revenue streams in the ecosystem.

The software structure is related to the design, development, and main-

[1]Social behavior of software denotes interaction between software elements, such as dependencies, data sharing, and interoperability

tenance of applications, services, and other types of software elements in the ecosystem. Part of the software structure is the software components or the software platform, while a software architectural description can be used to describe the software structure.

In the sections below, we use the concept of software ecosystem architecture and the three structures explained above to first describe healthcare in Denmark and then characterize the telemedicine ecosystem. We apply the structures of software ecosystem architecture to Danish healthcare, although we do not examine it as a software ecosystem. This is done to help understanding the telemedicine ecosystem better since it is heavily based on general healthcare. Thus the following chapter serves as background to the telemedicine ecosystem analysis.

3.1 Danish healthcare

Human health, although difficult to define, is claimed possible to be assessed by setting a number of parameters that identify a "healthy" condition or point to an "ill" condition (Schaeffer et al., 1988). The different parts and functions of the human body (and mind) can be interdependent to a great extent so that a condition on one part of the body can affect other parts. The treatment of human health, however, or rather the lack of health (i.e. disease or condition), does not seem to fully commit to the same principle. Healthcare services are compartmentalized into different sectors. These sectors, although they, as a general rule, accept the co-influence of each other, are only specialized in their own sector and tend to treat different conditions as separate instances only affecting the specific sector. Thus, a patient with diabetes that her condition has been affecting her nerve system and vision, should be treated separately by a dietician, a neurosurgeon, and an ophthalmologist, while changes in her condition observed by one specialist might not necessarily propagate to the rest.

This compartmentalization is also demonstrated by the task of the patient "hand over". Patients that need to be transferred across healthcare departments or healthcare professionals, have their information and control or responsibility handed over between health professionals. This process, although common, is many times criticized for lack of standardization and disregard to patient safety (Cohen and Hilligoss, 2010).

Naturally, the compartmentalization of healthcare in Denmark affects to a big extent the organizational structure of Danish healthcare, and this is also reflected to the business and software structures. In the following sections, we provide an overview of the three structures in Danish healthcare.

3.1.1 Organizational structure

The healthcare system is fairly decentralized and consists of three administrative levels: the state or government, the regions, and the municipalities. An overview of the organizational structure and administration of the Danish healthcare can be seen in Figure 3.1.

State level The Ministry of Health and Prevention governs the regional and municipal organization and management of healthcare. This state-wide

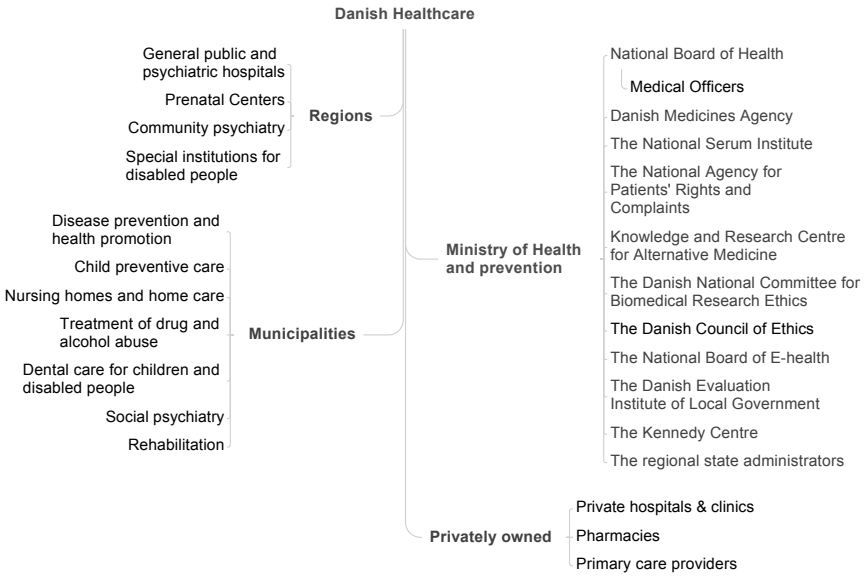


Figure 3.1: Danish healthcare organizational structure (taken from Olejaz et al. (2012)).

administration level governs in areas including medication, vaccination, in-vitro diagnostics, e-health, and biomedical research ethics.

Regional level There are five geographical regions each owning and running hospitals and finance private practitioners. They are responsible for the healthcare services provided in the region as well as the regional organization and funding.

Local level There are 98 municipalities that are locally responsible for prevention, health promotion and rehabilitation. Furthermore, the local level is also responsible for health-related services such as home care and care in nursing homes.

From the perspective of the provided services, the Danish healthcare system is divided into three sectors: the primary sector, that is responsible for preventive healthcare and practicing diagnosis and treatment, and the secondary and tertiary sectors that provide specialized care. The primary sector consists of private practitioners (e.g., general practitioners (GPs), specialists, and dentists) and municipal/local health service providers, while the secondary and tertiary sectors are mostly concentrated in hospitals. (Danish Ministry of Health and Prevention, 2008).

3.1.2 Business structure - financing

In the Danish welfare state, which serves as an example of the "universal model" (Esping-Andersen, 1990), citizens have access to a number of tax-funded services provided by the state. That is also the case with healthcare: most Danish

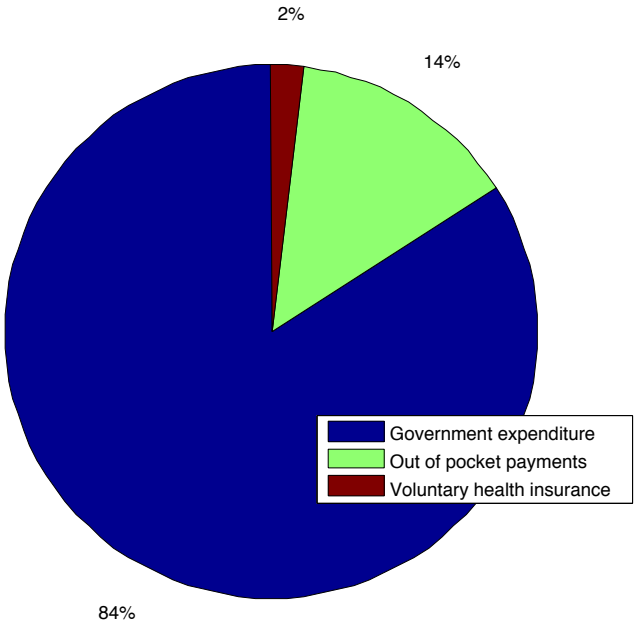


Figure 3.2: Danish healthcare expenditure by sources for 2007 (take from Olejaz et al. (2012)).

healthcare services are provided to the citizens without direct payment for the service they receive, since healthcare is funded by tax. This excludes private services not included in the state funding, such as the dental care and private hospitals. These count for less than 20% of the total health expenditure (Olejaz et al., 2012). The services included in the state funding are funded from two different kinds of taxes related to healthcare: the municipality tax, paid to the municipalities (that is, among others, funding the municipalities' healthcare activities) and the "health contribution" tax paid to the state. The regions, that are the main healthcare provider is funded by 80% from the state and 20% from the municipalities (Andersen and Jensen, 2010). Their funding is a combination of block grants (86% in 2011) and activity funding (14% in 2011) (Olejaz et al., 2012).

Figure 3.2 provides an overview of the sources for the Danish healthcare expenditures in 2007. Government expenditure covers the biggest percentage of healthcare expenditure, while there is a small percentage of citizen expenditure, mainly for dental services, medication prescribed outside the hospitals, and glasses. Citizens can obtain a voluntary health insurance to cover some of their expenditures (Olejaz et al., 2012).

3.1.3 Software structure - digitalization

Denmark is considered one of the leading countries in health IT (Bhanoo, 2010; The Economist, 2011). The primary healthcare sector with the general practitioners (GPs) has a high level of digitalization and standardization of communication. Since 2004 all GPs use systems connected to a national network that

enables them to send and receive electronic clinical data to and from a number of healthcare actors among them specialist doctors, hospitals, pharmacies and other GPs (Protti and Johansen, 2010). In 2010, 90% of the clinical communication between primary and secondary sector was electronic (Olejaz et al., 2012).

The secondary sector has a wider range of actors, specializations, and types of information to be managed. There are a number of systems and applications that support and promote the exchange of healthcare-related information, although not necessarily specific to the secondary sector. Below we provide three examples of these systems.

Sundhed.dk The portal Sundhed.dk is the "official portal for the public Danish Healthcare Services" aiming to enable "patients and healthcare professionals to find information and communicate" (Sundhed.dk, 2014). Sundhed.dk is managed by the ministry of health and prevention, the five regions, and KL, the interest group and member authority of the Danish municipalities.

Shared Medication Record (FMK) An online service that aims at providing citizens and healthcare professionals with access to citizen medication and vaccination (SSI, 2014a).

Healthcare Data Network (SDN) A secure communication network enabling the exchange of patient information between healthcare actors such as hospitals, GPs, laboratories, and pharmacies. It was established in 2003 and is managed by the state-controlled information standardization organization MedCom (MedCom, 2014a).

Although Denmark demonstrates a high level of digitalization in healthcare, things are far from perfect. Danish healthcare IT systems are characterized by lack of coordination (Rudkjøbing et al., 2012) and fragmentation (Kierkegaard, 2013). Healthcare IT systems in this country of roughly 5.6 million inhabitants is divided into five administrative regions having implemented four different electronic medical records (EMRs) (Kierkegaard, 2013), while critique on the current management of healthcare is that "perverse financial incentives" are "barriers to integration" (Rudkjøbing et al., 2012). This is also supported by the expensive record on failed healthcare digitalization initiatives such as G-EPJ and the National Patientindex (Kierkegaard, 2013; Aanestad and Jensen, 2011; Lund, 2013).

3.2 The Danish telemedicine ecosystem

In this section, we characterize the Danish telemedicine ecosystem in relation to the healthcare ecosystem both before the influence of Net4Care and 4S, refereed to as the "current telemedicine ecosystem", and after the influence of the projects and towards the establishment of a software ecosystem. We define the Danish telemedicine ecosystem as the actors, (software and hardware) components, services, and their relationships evolved from the telemedicine services of Danish healthcare. Telemedicine can be defined as the "delivery of health care services, where distance is a critical factor, by all health care professionals using information and communication technologies" (World Health Organization,

2010). The telemedicine, ecosystem, therefore, is an ecosystem that provides solutions (products or services) aimed at facilitating the prevention, diagnosis, and treatment of patients.

In the following sections, we use the three structures of software ecosystem architecture and expand on the telemedicine-specific particularities of the healthcare analysis.

3.2.1 Current telemedicine ecosystem

Organizational structure

In order to define the organizational structure of the ecosystem, we analyze the actors of the ecosystem from two perspectives: the *actor structure* and *actor function*. We are inspired by research on natural ecosystems where Schaeffer et al. (1988) separate between the different ecosystem measures to measures of structure and function. Measures of ecosystem structure measure elements that affect the structure of the ecosystem such as the number of species that form part of the ecosystem and the kinds of organisms, while measures of functions focus on measures that affect the function of the ecosystem such as the activity or production of the species.

The actor structure of the telemedicine ecosystem largely follows the actor structure the general healthcare. Most of the organizational actors identified in Section 3.1.1 such as the ministry, the regions, or the municipalities, are also organizational actors in the telemedicine ecosystem. Naturally apart from the organizational actors, the ecosystem includes actors developing and maintaining technical systems and infrastructures or actors responsible of the project management. We identify the following actor roles:

Developing actor. Actors that are involved in the ecosystem with a specific task. Tasks such as application development, application maintenance, application support, project management, or supply of related assets or services.

Host. Actors that are responsible for hosting telemedicine applications. This kind of organizations typically represents the product owner and customer or end user.

Orchestrator. Actors involved in the governing body of the ecosystem typically responsible for the technological platform(s).

If we look at the actor function in the ecosystem, we separate the actors into *keystones* or *dominators* (Iansiti and Levien, 2004b). A keystone is an actor that is essential to the ecosystem and potential extinction of a keystone would have consequences for several actors or even the whole ecosystem. The keystone supports the prosperity of the ecosystem through actions that result in the benefit of other actors or the whole ecosystem. A dominator, on the other hand, is similar to a keystone but has the tendency to grow in size by eliminating surrounding species. The roles of the eliminated species in the ecosystem are then either taken over by the dominator or disappear from the ecosystem. The dominators are harmful for the ecosystem's health as they reduce the ecosystem diversity and thus affect niche creation. In Paper 5 we characterize the function of the most influential actors of the telemedicine ecosystem and show that some can have both keystone and dominator functions. Moreover, we observe the tendency of having several actors involved in only one application (specialized actors). That is a sign of high diversity in the ecosystem that supports niche

creation and could promote the health of the ecosystem.

Additionally, another point that affects the organizational structure of an ecosystem is the ability of the ecosystem to introduce new actors (i.e. how open or closed the ecosystem is to new actors). In order to analyze how open the ecosystem is to new actors, we make a separation between the three actor roles: developing actors, hosts, and orchestrators.

The involvement of developing actors is done, as we discuss in Section 3.2.1, either with a call for tenders where the actor with the accepted tender is appointed for the required task, or for services with cost lower than 500,000 Danish crowns through direct contracting. The ecosystem is relatively closed to external developing actors: the ecosystem allows new developing actors to be involved but the new actors are subjected to an acceptance rate (usually one out of the applicants) and following a procedure of submitting a tender that might prove time and resource demanding.

The ecosystem is closed to new host organizations: there is currently a number of hospitals and municipalities that can be used as hosts in telemedical projects. Orchestrators can decide to include other organizations as hosts if there is a need. Moreover, in some cases, a developing organization might also serve as an application host. In that case, the ecosystem is almost as open to hosts of this kind as to developing organizations, but with the additional restriction that the host should commit to privacy and security regulations concerning healthcare data. Traditional hosts such as hospitals are also committed to these regulations as this is part of the everyday work in a hospital.

The ecosystem seems to be completely closed to orchestrators with rare exceptions. As we discuss in later sections, since the ecosystem does not have one common technological platform, the assessment of how open the ecosystem is to organizations of this role is not possible. Orchestrators of existing platforms that are not specific for the ecosystem or widely used in the ecosystem (e.g. National Service Platform and Healthcare Data Network) have been introduced by appointment of state level organizations but these platforms are few and this seems more like exception to rule rather than a systematic inclusion of actors.

Business structure

Using Section 3.1.2 as a background, we note that in the telemedicine ecosystem the most common way of developing an application or system is to have a project initiated that addresses a specific issue. In such a project, the orchestrator is mostly represented by the region(s) or the state, the users (or part of them) by one or several hosts and the project is developed by one or several developing actors. The activities and actors are funded by the involved regions while the end users (patients, clinicians) are external to the funding process.

Following these revenue streams, the telemedicine ecosystem actors demonstrate symbiotic relationships both among themselves but also with the ecosystem. The regions and the state have a specific amount of money to invest and it is up to management (or orchestrators) to delegate the funding to the appropriate projects. A project getting over budget, or a hospital investing in implementing a premise-specific solution to a problem that is faced by other actors can be characterized as dominator activity, since it is reserving funding that could be used to the benefit of more actors.

At the same time all actors benefit from the well functioning and success

of the ecosystem as a whole. The telemedicine ecosystem might seem robust as it is funded by the state. However, private healthcare is increasingly taking over services from the public with the private hospital activities increasing by 2% over the total healthcare activities in the years from 2002 to 2010 and the number of patients treated in private hospitals increasing by 134,721 for the same years (Sundhedsstyrelsen, 2011).

Software structure

The software structure of the telemedicine ecosystem is characterized by hundreds of uncoordinated, small projects, each developing their own solution, including infrastructure. Those solutions are most often not able to share data, due to the lack of common infrastructure, and they most often disappear when the resources of the project are consumed. This has resulted in more than 350 current telemedicine initiatives (MedCom, 2013) of which the minority are in production. This is also demonstrated in Paper 5 where we graph the relationships between actor and telemedicine application for the Capital Region of Denmark. This study reveals a set of sparsely allocated clusters with low connectivity among them. Furthermore, it shows that on average one application connects to less than one (0.72) application, while the most influential applications (according to their node degree) usually do not connect. This view of low interoperability is also supported by the project based call for tender involvement of developing actors, where different actors are involved according to their ability to perform a task creating specialized solutions.

When examining the platform of the ecosystem, we note that Danish healthcare includes common technological platforms for secure communication, video communication, and service-oriented computing (the "Danish Healthcare Data Network" (SDN) (MedCom, 2014a), the "Healthcare Video Hub" (VDX) (MedCom, 2014b), and the "National Service Platform" (NSP) (SSI, 2014b) respectively). These platforms have, however, not been built for purposes of telemedicine (with patients directly involved in use), and thus, we notice the lack of a widely used, common platform for telemedicine applications. Applications might be using the same systems (like identifying patients through their civil registration number) but the solutions are mainly ad hoc, which points to the problems in the development and implementation of telemedicine applications investigated in this thesis.

Not (yet) a software ecosystem

Examining the characterization of the telemedicine ecosystem, we note that it deviates from the definition of a software ecosystem of Chapter 2. That is mainly because the software structure, as noted in the previous section, lacks a common technological platform while the existing platforms in the telemedicine ecosystem are not telemedical platforms per se. This is a big part of the problem that this PhD project is focusing on. In this thesis we focus on the telemedicine ecosystem specifically, as it is defined in the beginning of this chapter, and do not examine telemedicine as part of the general healthcare services, because we identify that the domain of telemedicine has some particularities separating it from the general healthcare domain. However, we argue for the telemedicine ecosystem being an ecosystem since the actor structure commits to an ecosystem

structure as they have symbiotic relationships among them, while their survival in the ecosystem depends on the survival of the ecosystem as a whole.

3.2.2 Towards a Danish telemedicine software ecosystem

During this PhD project the current telemedicine ecosystem was characterized, as noted in the previous section, by the lack of a common technological platform. This was part of the problem that the Net4Care project was focusing on. The Net4Care project, investigated the design and development of a common platform that would address three main issues in the development of telemedicine applications: lack of integration, low buildability, and lack of reuse. The output was the Ne4Care platform (Net4Care, 2014) that was based on the following architectural decisions:

- *Information resources* primarily in the form of open source well proven and tested tutorials on the web site `http://www.net4care.org` to shallow the learning curve for developers.
- *Reference implementation* as open source.
- *Staged testing environment*, which provides a simple isolated test environment for fast development that seamlessly can be migrated into a full operational environment.
- *Clinical standards* used at the back tier for storage format (Continua Alliance *Personal Health Monitoring Record* (PHMR) (HL7, 2010) which is a HL7 "Clinical Document Architecture" standard (Boone, 2011)) as well as for database system (XDS.b *Cross-Enterprise Document Sharing* (IHE, 2013) which is a standard for clinical storage systems).

The establishment of Net4Care as a common infrastructure for the implementation of telemedicine application would change the software structure of the ecosystem. Telemedicine applications would be developed on top of Net4Care, thus the software dependency graph would be clustered around the platform. Additionally, the platform would take care some of the common functionality, like storing healthcare data, that would be normally handled by each telemedicine applications separately.

The Net4Care platform would bring the telemedicine ecosystem closer to a software ecosystem. However, in order for the telemedicine ecosystem to become a software ecosystem, apart from changes to the software structure (with the implementation of a common platform), the organizational and business structures have to be adapted. The establishment of a common platform would require an actor or a set of actors to manage it and serve as additional orchestrators. This proves to be a challenging task due to the fragmentation in the organizational structure: in order for the platform to be the common platform of the ecosystem it should be accepted by all the existing orchestrators in the ecosystem (such as state and regions), while it might be hard to reach a consensus between the orchestrators on who to manage the platform. At the same time, the organizational and business structure did not provide incentives to developing actors to use the platform unless the platform became the established common ecosystem platform. These issues gave ground to the establishment of the 4S organization,

an organization that would take the role of an orchestrator responsible for the following tasks:

- *Platform management.* Be responsible for managing the platform, maintaining it in a state that would continue to provide value to the actors and the ecosystem as a whole, and commit to reflecting the governance applied from the rest of the ecosystem orchestrators (e.g. state, regions).
- *Promote healthy ecosystem organizational characteristics.* Support ecosystem variability and robustness reflected from the actor and software activity.
- *Invest in community building.* Support the establishment of a community around the telemedical ecosystem including domain actors such as doctors and medical care personnel and thus supporting the ecosystem robustness and productivity.

3.3 Summary

In this chapter we describe the telemedicine ecosystem. We follow the approach of software ecosystem architecture and the three structures that model a software ecosystem, proposed in Paper 6: organizational, business, and software structure. We identify that the telemedicine ecosystem, since part of the Danish healthcare, has a big overlap of structures with the Danish healthcare. Therefore we initially analyze the organizational, business, and software structures of Danish healthcare that are relevant to the telemedicine ecosystem and in continuation we analyze the telemedicine ecosystem. We do so by analyzing the telemedicine ecosystem in two snapshots: the current ecosystem as it was before the influence of the Net4Care project and 4S organization (and, thus, also the influence of this PhD project) and after, towards a software ecosystem. We describe the current telemedicine ecosystem as a relatively closed to external actors ecosystem, that mainly consists of three actor roles: developing actor, host, and orchestrator. We identify high actor diversity that could tentatively indicate ecosystem robustness as a means of preserving ecosystem health. We find that the business structures support our findings of the organizational structure and we identify that although funded from the state and thus robust, the ecosystem is facing an increase of private healthcare that might challenge the ecosystem probability of survival. Furthermore, we identify that the ecosystem is characterized by the lack of a common technological platform, a fact that makes the ecosystem deviate from the definition of a software ecosystem.

When examining the influence of this project to the telemedicine ecosystem, we identify that the Net4Care project is addressing the lack of platform and serves as the first step towards the establishment of a software ecosystem. The 4S organization can be considered as a following step, by managing the platform and focusing on the lack of social networks and communities in the ecosystem in a way that would promote the health of the under establishment software ecosystem.

Chapter 4

Conclusions

In this chapter we conclude on the work of this PhD project. To do so, we summarize the contributions of this project separating them into contributions to the theory of software ecosystems and to the domain of the telemedicine ecosystem. We discuss aspects not analyzed and limitations of our studies, elaborate on future work, and conclude.

4.1 Contributions

4.1.1 Software ecosystems

The field of software ecosystems is the main theoretical domain that this project is based on. Thus, a main part of the contributions of this project is to this domain. This is also indicated by the included papers as all of the papers contribute to the field of software ecosystems. Our contributions to this field cover several aspects of software ecosystems. From one side there are contributions to general aspects of software ecosystem such as providing an overview, defining the field, and proposing means to analyze or construct whole ecosystems. While from the other side, there is focus on more specific aspects such as defining ecosystem health, measuring the governance influence to health, graphing social networks of actor and component interaction, and identifying important actors or software components. Below, we explain the contributions for each of the included papers.

Paper 1

Paper 1 is the first and only, according to our knowledge, systematic literature review of the whole field of software ecosystems. In this study, we review the published literature on software ecosystems up to June 2012. In total we review 90 papers out of a gross list of 420 according to our review protocol that follows the guidelines of Kitchenham and Charters (2007). This study provides an overview of the field. It contributes to the field of software ecosystems by identifying and analyzing the different definitions of software ecosystems and the different groupings or perspectives in the definitions, while proposing a comprehensive definition that includes the different aspects of software ecosystems. Furthermore, it proposes analyzing software ecosystems by using the concept of

'software ecosystem architecture' and identifies three logical groups in software ecosystems: software engineering, business and management, and relationships. Finally, it identifies that almost half of the literature does not study existing software ecosystems, while the majority of the papers studying existing software ecosystems are examining open source ecosystems. This implies that they are having a more technical and possibly social focus and potentially excluding business perspectives and applying somewhat different governance approaches.

Paper 2

Paper 2 contributes to addressing the lack of empirical studies identified in the literature review by studying two software ecosystems in Apache Maven Repository, the repository of Maven, a build automation tool. In this study we investigate the dependencies and interaction between the software elements of a software ecosystem, and graph them as a network of software elements. To analyze this social network, we take inspiration from the analysis of natural ecosystems as graphs through "network ecology" (Jordán and Scheuring, 2004) and investigate software ecosystems using metrics from network ecology. Furthermore, we use the PageRank (Brin and Page, 1998) algorithm to study the structure of software ecosystems. We define the concept of an ecosystem "neighborhood" and study each of the two ecosystems according to their neighborhoods. Finally, we propose the use of a "keystone index" and PageRank as means of identifying the most influential software components in an ecosystem and provide the means for comparing two similar ecosystems.

Paper 3

In Paper 3, we continue on the idea of measuring software ecosystems by focusing on establishing measures for the quality of the ecosystem. In particular, we focus on the concept of "health" of a software ecosystem. In this study we review the software ecosystem health literature and, using the "snowballing technique", we review the literature that inspired the software ecosystem health literature. In total, we review 23 papers from four different fields: software ecosystems, business ecosystems, natural ecosystems, and open source software. We identify that the majority of the software ecosystem literature (11 out of the 13 papers) is inspired by the health of business ecosystems and more specifically the definition of Iansiti and Levien (2004a,b,c) and Iansiti and Richards (2006), who in their turn are inspired by natural ecosystems. Moreover, we identify two main differences between software ecosystems and business and natural ecosystems, as they appear in our literature: the fact that the actor is not the product per se and that there usually exists an orchestrating entity. Furthermore, as the main contribution of this work, we propose a conceptual framework for the definition of health in software ecosystems that consists of three components: 'actors', 'software', and 'orchestration'. The actor influence over the ecosystem health is by both the collective health of each actor and the influence of the actor interaction network. Similarly, the software influence over the ecosystem health is by the collective health of each software element and the influence to the health of the software network of interaction. In the case of software, however, there is a software element that has special influence to the ecosystem health: the common platform. The common platform is highly influential, as by definition

connects to all the software products, and can serve as means, but not the only, for the orchestrator to apply orchestration acts.

Paper 4

In Paper 4 we focus on how software ecosystems are governed. We apply the governance model proposed by Jansen and Cusumano (2012, 2013) to the Application Development Partner (ADP) software ecosystem evolved around the products of Axis, a network video and surveillance camera producer. Our initial contribution is to define a "hardware-dependent" ecosystem as the ecosystem where hardware plays a dominant role in the value creation process and where the customers purchase hardware devices with software installed on them. To the best of our knowledge, no previous study has evaluated the governance of a hardware-dependent software ecosystem. Our results reveal that although the governance actions do not address the majority of the applied governance framework of Jansen and Cusumano, the ADP ecosystem is considered by its participants as a growing ecosystem providing opportunities for its actors. This could be explained by the fact that Axis, as the orchestrator and the platform owner, does not address productivity and robustness of the ecosystem, but has a network of vendors and resellers to support it. Moreover, several of the governance activities (e.g. communication) are achieved by non-formal means. We contribute to the study of software ecosystem governance by identifying perspectives that the existing governance framework does not cover. This serves as input to improve the framework in order to cover a wider range and different aspects of software ecosystems and come closer to a more efficient evaluation of the ecosystem governance in a way that would support the health of the ecosystem.

Paper 5

Focusing more on the applied domain of this project, Paper 5 is studying a part of the telemedicine ecosystem, the telemedicine services of the Capital region, the biggest in terms of population of the five healthcare regions of Denmark. In this study, we characterize part of the telemedicine ecosystem by studying the interaction of actors and telemedicine applications (software). We are inspired by the influence the actor and software networks have in the ecosystem health as analyzed in the health framework from Paper 3 and use the PageRank algorithm to measure the importance of nodes (actors and software components) applied in Paper 2. We establish a method for this study, where we define two actor roles specific to the ecosystem and ways of characterizing actor contributions. To our knowledge, there is no previous study that conducted a quantitative analysis of the actors of a non open source software ecosystem. Related work shows social analysis of open source ecosystems but we argue that the social networks of actors and software of proprietary ecosystems are different. Our study revealed, by analyzing the social networks of the ecosystem, that it is possible to deduct essential information about the ecosystem functioning that, if addressed properly, might lead to better supporting the health of the ecosystem.

Paper 6

Finally, in Paper 6 we focus on how to analyze existing software ecosystems but also how to design new ecosystems. We are inspired from Paper 1 and expand the definition of "software ecosystem architecture" as the "set of structures needed to reason about the software ecosystem, which comprise actor and software elements, relations among them, and their properties". We identify three structures as essential to reason about the software ecosystem: organizational structures, business structures, and software structures and identify the main elements and their relationships for each structure. Furthermore, we demonstrate how this concept can be used to analyze an existing ecosystem and to design a new ecosystem. We analyze an existing ecosystem by applying the architecture to the Danish telemedicine ecosystem. Moreover, we design a new ecosystem where we analyze the ecosystem under-development around the Net4Care and OpenTele platforms and the 4S organization as an orchestrator.

Concluding theoretical contributions

Concluding our contributions to the field of software ecosystems we examine the research questions related to the theoretical domain. To address Research Question (*ii*), *"How can the concept of 'software ecosystems' be defined, and how can software ecosystems be analyzed and modeled?"*, we take the following steps: (a) we define software ecosystems by analyzing all the definitions in the literature, including the different perspectives of the definitions and (b) we propose the analysis and modeling of software ecosystems using the concept of software ecosystem architecture, which consists of the organizational, business, and software structures. We demonstrate the use of our concept by applying it first to the telemedicine ecosystem and then demonstrating the steps towards the establishment of a software ecosystem in the telemedicine ecosystem. Part of the analysis of the organizational (and indirectly software) structure includes the study of the (social) interaction between actors, software, and between actors and software. This analysis is initially inspired by natural ecosystems and the theories of network ecology and we have first applied them on the open source ecosystems of Apache Felix and Eclipse Equinox.

In order to address Research Question (*iii*), *"How can we define and measure the qualities of a software ecosystem?"* we take the following steps: (a) we identify the use of the concept of 'software ecosystem health' in the literature as means of describing the quality of an ecosystem. (b) We conduct a literature analysis of the health in software ecosystems and fields that inspired the software ecosystem health. As a result, we propose a framework for defining the health of a software ecosystem. (c) We identify software ecosystem governance as having influence on the ecosystem health and apply an existing governance model to a hardware-dependent ecosystem with particularities in the orchestrator business models. Our conclusions identify areas where the existing model could improve. Concluding Research Question (*iii*), the quality of a software ecosystem can be expressed as the health of the ecosystem, influenced by the health of each individual actor, the actor interaction, each individual software component, the component interaction, the common platform, and the orchestration. Moreover, an aspect influencing orchestration and, thus the health of the ecosystem, is the governance of the ecosystem and the extent that it promotes healthy activity.

This project has a number of additional contributions to the theoretical field that is not part of the included papers. In Syeed et al. (2014) we study the Ruby ecosystem, the open source ecosystem developed around the Ruby programming language, to reveal possible socio-technical congruence, i.e. whether there is a correlation between the coordination of the ecosystem actors and the coordination of the software components. Our findings reveal that there is a low congruence in our ecosystem while our interpretation of the findings is that the ecosystem is too big for the actors to have intensive communication outside their projects scope.

Moreover, in Manikas et al. (2014) we examine app store governance mechanisms for mission critical (our case being healthcare) apps. In this study we study four cases of app governance in the healthcare field and propose a model for the analysis of apps consisting of three categories: aim, impact, and revenues, We apply our model in apps of the healthcare and wellness domain and propose three governance mechanisms according to the assed risks the apps pose.

4.1.2 Danish telemedicine

The problem that this project is investigating is positioned in the Danish telemedicine, the applied domain of this thesis. The contributions in the theoretical domain of software ecosystems also related, directly or indirectly, to the application of software ecosystems to the Danish telemedicine. Below, we explain the main contributions to the applied domain. Two of the included papers, Paper 5 and Paper 6, have direct contributions to the telemedicine ecosystem while we also describe shortly the contributions of the Net4Care project and 4S initiative.

Paper 5

As explained in the previous section, in Paper 5 we characterize the Danish telemedicine ecosystem. To do so, we explain the ecosystem by defining the ecosystem domain and borders and analyze its main elements: the set of actors and organizational structure of the healthcare in Denmark that is applicable to our ecosystem, the business models and financing of the activities of the ecosystem, and the software elements. We note that the ecosystem, at the time of study, is characterized by the lack of a common technological platform, thus not falling under our definition of a software ecosystem. However, we argue for the use of software ecosystem theories in analyzing the telemedicine ecosystem for two reasons: (i) the actor structure commits to a software ecosystem actor structure both in terms of symbiotic relationships, business models motivating them and having as products software and services and (ii) we identify that part of the problems in the telemedicine ecosystem is due to the lack of a common and widely used technologic platform. Thus, studying the telemedicine ecosystem as a software ecosystem points to issues for improvement in the ecosystem.

We analyze the telemedicine actor and software interaction of the Capital healthcare region of Denmark. In total we graph 28 applications, 109 actors, and 182 connections among them. Our graph shows clusters around the telemedicine applications with low connectivity among the applications (0.72 per application), while the biggest, in terms of node degree, applications of the graph are not connected. Moreover, we noted that each actor is mainly connected to one application (1.52). This implies specialized actors, actors with a specific

task in the ecosystem, and denotes a high level of variability thus promoting the ecosystem health. Furthermore, when looking at the most influential actors we discover activity both beneficial and harmful for the ecosystem. The organization MedCom, a non-profit, state-established and -funded organization has the role of creating standards, project management and quality assurance of telemedicine applications, thus providing value to the surrounding actors by supporting and assisting their work. This activity is characterized as beneficial (keystone) to the ecosystem. On the other hand, Hvidovre hospital, is the most influential host actor and has among its hosted applications three applications for which the hospital is the only host and that seem to be developed specifically for that setting. This is characterized as harmful (dominator) activity for the ecosystem as this actor is reserving common funds for local implementation of a solution to a problem faced by several host actors.

Paper 6

In Paper 6 we expand on the work of Paper 5 and analyze the telemedicine ecosystem as a whole. We do so by using the approach of the software ecosystem architecture and investigate how to analyze existing ecosystems and design new, as explained previously. In this study we analyze the organizational structure of the telemedicine ecosystem by identifying the main actors of the ecosystem, defining three actor roles specific to the ecosystem, and analyzing how open the ecosystem is for each of the actor roles. Furthermore, we analyze the business structure of the ecosystem by explaining the process of actor involvement to the ecosystem and analyzing the business model of a popular telemedicine project. Finally, we analyze the software structure by analyzing the existing systems and identifying the main issues in the existing structure.

In continuation, we analyze the design of a software ecosystem around the telemedicine ecosystem using again the approach of software ecosystem architecture. In the software structure there will be the Net4Care platform supported by the OpenTele, while in the organizational structure the establishment of the 4S organization will serve as an orchestrator of the ecosystem managing the platform and enabling actor involvement. Moreover, we explain how the business models around the telemedicine ecosystem are modified. In the following sections, we explain the contributions of the Net4Care and 4S in more detail.

The Net4Care project

A contribution that is not strongly highlighted in the included papers, is the Net4Care project. The Net4Care project was initiated focusing on some of the same problems behind this PhD project: the technical challenges behind the development and the adoption of telemedicine technologies in Denmark. More specifically, it was focusing on three issues identified in the telemedicine systems: lack of integration, low buildability, and lack of reuse. Investigating the approach of building a software ecosystem, we initiated by designing an infrastructure that would serve as a common technological platform to facilitate the evolution of a software ecosystem. Some of our scenarios included ways of enabling different actors for the ecosystem and we focused especially in enabling small and medium businesses (SMBs) to contribute to the ecosystem under construction as means of increasing ecosystem productivity, fostering innovation,

and enhancing the popularity of telemedical application development. The main focus of the platform design can be summarized as (i) the option of a reference implementation instance for easier client development, (ii) the existence of informational resources by means of "hello world" examples, how-to's and tutorials to shallow the learning curve of new developers, (iii) isolated test environments, and (iv) use of existing and common clinical standards. The outcome of this study is the Net4Care open source platform both implemented and as a reference implementation along with a set of how-to's and documentation[1]. The platform has been evaluated by a number of actors that have been exploring and contributing to the different aspects[2]. The specific contribution of this PhD in the Net4Care project is that I have been involved as a member of the project contributing to the pre-analysis, design, development, implementation, and evaluation of the platform.

The 4S initiative

The establishment of a technological platform, although essential, might not necessarily be adequate for a software ecosystem to emerge. Especially in the case of a newly established platform that is not yet widely recognized. That is the case with Net4Care. In order for a software ecosystem to emerge in the Danish telemedicine ecosystem, the organizational and business structures have to be better supported to promote the emerging of a software ecosystem. That is the aim of the 4S organization. It has as a purpose to foster a software ecosystem in the telemedicine services of the Danish healthcare. 4S is focusing in providing orchestrator services in the software ecosystem under development. It has taken the task of organizing and managing the technological platform (Net4Care and the expansion to include OpenTele) both technically but also coordinating and supporting the network of actors developing on the platform. Moreover, it has expanded its activity in supporting the social network of actors by (i) obtaining commitment from interested traditional orchestrating actors (such as state, regions, municipalities), (ii) involving end-user such as patients and clinicians, and (iii) involving developing actors by supporting the accessibility of the platform and thus, the ecosystem. Furthermore, 4S is supporting the business structure by providing value proposition and viable business models for developing actors, orchestrators and hosting actors. This PhD has contributed to the 4S organization both by being part of the Net4Care, the technological platform to be orchestrated by 4S and by providing input to the 4S organization in issues related to supporting and promoting the ecosystem health and governing the ecosystem.

Concluding domain contributions

Concluding our contributions to the applied domain of Danish Telemedicine, we explain to what extent we address Research Question (*i*), *"How can a software ecosystem foster the development, implementation, and use of telemedicine services?"*. Initially we characterized the telemedicine ecosystem and identified the lack of a common platform and limited actor and software interaction. This confirms our initial hypothesis that the establishment of a software ecosystem on

[1] Platform and documentation available under `http://net4care.org/`

[2] More information about the evaluation on Paper 6.

the telemedicine ecosystem could address the challenges currently faced by the telemedical technologies, as a software ecosystem implies a common platform and symbiotic relationships between actors and software. Evaluating what are the qualities of a successful ecosystem in general and for the telemedicine ecosystem, we designed, developed, and implemented Net4Care to serve as a common technological platform that would support increased integration, high buildability, increased reuse, and thus, support (stronger) interaction of software and, indirectly, actors. Furthermore, we provided input on how to establish 4S, an organization that would promote the health of the telemedicine software ecosystem under establishment through the proper governance, platform management, better value proposition, and support of social interaction.

Concluding, we address Research Question (i) by identifying the need and taking the steps towards a software ecosystem with (i) a common platform that supports and promotes the development of different actors on top of it, (ii) increased software interaction, (iii) stronger social network of actors, (iv) robust business structures better supporting the actor involvement in the ecosystem, and (v) better orchestration and governance of the ecosystem to promote and support these changes and the health of the ecosystem.

4.2 Discussion

From this project, we note that the problems inhibiting the development of telemedicine technologies, although examined from a rather technical aspect, do not find solution solely in one theory or theoretical domain, but in a set of different theories. This is nicely suited with the multidiscipline aspects of software ecosystems that interweaves technical, organizational, social, and business perspectives. Naturally, any changes to the healthcare services, especially in a state-sponsored establishment of healthcare services, can have a wide impact on the provided services possibly on a country level. This is a particularity of the studied ecosystem, that has not been explicitly explained in the thesis: since the telemedicine ecosystem is part of the country-wide provided healthcare services, the public opinion and entities relating to the public opinion, such as political parties and media, can have an indirect influence to the governance and functioning of the ecosystem.

Moreover, the software ecosystem under development is bound to the particularities of the healthcare services of Denmark. There are several countries that follow some of the same characteristics of healthcare, such as organizational structure and financing models, that our software ecosystem could be adopted or expanded to. For example, one of the main particularities of this ecosystem is the fact that the majority of the healthcare activities are funded by the state (with 84.5% of the total healthcare expenditure, (Olejaz et al., 2012)). There are a number of countries that follow the same level of involvement of the state in the provided healthcare services, such as Norway (85%) (Ringard et al., 2013), Sweden (83%) (Anell et al., 2012), or France (79%) (Chevreul et al., 2010). Furthermore, it might be relevant to examine the adaptation to countries with different organizational structures and financing models. An example is the case of Greece, that is one of the countries with the largest shares of private health expenditure in Europe, with 39.7% of the total health expenditure (Economou, 2010). Telemedicine in Greece is suffering from some of the same problems as

the telemedicine ecosystem under study: a set of separate projects with lack of continuity and not "integrated into a coherent governmental policy" (Bamidis et al., 2006). Problems that, as we have shown in this thesis, could be addressed with the establishment of a proper software ecosystem.

Additionally, our study is bound to a set of limitations, one of them being the maturity of the theoretical domain. During the initiation of the project, the theoretical domain of software ecosystems was immature, as we also noted in Paper 1. This implied that there was low or no experience with studying ecosystems similar to our ecosystem. It is indicative that there was no study of mission-critical software ecosystems before the start of this project. Therefore, many theories had to be either built from new or tested in our ecosystem settings, thus, compromising the external validity of our studies. We acknowledge, however, that during the project the field has evolved and matured, our work being a part this evolution. This maturity is also indicated by the increase of studies reporting empirical models within the last two years noted in Section 2.1.2.

An additional limitation of the theory is the polarization between open source ecosystems and proprietary ecosystems. The field of software ecosystems is inspired by and combines characteristics of both open source and proprietary software development and many times a software ecosystem can be characterized as a hybrid of the two. For example, the use of a community or network of actors or the development of software by external actors on top of a platform are practices found in open source projects, while the monetization of the developed products is something primarily found in closed source or proprietary development. A widely known example of a hybrid ecosystem is the Android ecosystem, the software ecosystem build around the smart device operating system orchestrated by Google. The platform, Android, is offered as open source, while the apps that are built on top of it, and distributed through the Google Play app store, are mainly proprietary. However, results and conclusion from studies of purely open source or proprietary software ecosystems, although beneficial, might not be applicable to diametrical ecosystems. For example a study about the developer contribution in Gnome as appeared in Goeminne et al. (2013) is hard to apply to our telemedicine ecosystem, a hybrid, in similar lines of the Android example, with an open source platform and possibly proprietary products on top of it. Studies of open source software ecosystems are popular in the research community as open source projects provide access to detailed information, like source code, mailing lists, or bugs, as opposed to proprietary ecosystems where much of this information is considered asset and cannot be revealed to third parties. However, pure open source ecosystems are usually characterized by a different business structure as the actors are not necessarily involved in the ecosystem for monetary incentives. This affects both the software structure and the actor relationships. Since there is no monetization of the produced software, source code is not protected making it open and for anyone to modify and reuse, depending on the software license. This also affects the actor relationships as the actors tend to interact more with other actors of the ecosystem.

This difference between open source and proprietary software ecosystems was identified early in our project and expressed in Paper 1. It was also pointed out as one of the contributions of Paper 5 where it was the first study, according to our knowledge, to study the social network of actors of a non open source

ecosystem. In the studies of this project we focus on conducting studies to open source ecosystems with results that would be generalized to non open source ecosystems. An example is the use of PageRank applied in the study of Eclipse Equinox and Apache Felix, two open source ecosystems studied in Paper 2 that was then applied to the actors and applications of the telemedicine ecosystem in Paper 5.

4.3 Future work

This project is the first step in demonstrating that it is possible to create an ecosystem by identifying the need for one. In order for an ecosystem to be successful, however, there is more required than just a common platform. Examples like the failed Symbian ecosystem of Nokia (Null, 2013) support this argument and underline that there is not adequate knowledge on how to successfully manage software ecosystems, thus making existing ecosystems hard to predict and control and new ecosystems hard construct. For an ecosystem to survive and remain productive over time, a proper orchestration strategy is required. Orchestration of a software ecosystem is the activity, i.e. use of tools, governance and management of the elements of an ecosystem, which results in influencing the ecosystem as a whole in a controlled manner. Future work of this project would include investigating the properties of successful orchestration, i.e. the orchestration that promotes the activity of an ecosystem maintaining the ecosystem variable and productive over time, and evaluate the identified properties by implementing them in the emerging ecosystem of the Danish telemedicine.

Furthermore, app stores are becoming a necessary component of several software ecosystems, since they provide seamless integration of externally developed applications into the common platform and a centralized and collected way of distributing software applications. Future work thus includes the expansion of our work published in Manikas et al. (2014) and investigate the parameters of building an app store for the Danish telemedicine ecosystem: an app store with high requirements on security and governance, for an ecosystem of high-risk mission-critical software with potentially high rate of app production.

4.4 Summary

Denmark's demographic changes and the organizational and financial changes of the Danish healthcare are challenging the levels of provided healthcare and point towards the establishment of telemedicine as means of patient support, diagnosis, and treatment. However, telemedical solutions are faced with a number of challenges that inhibit their adoption, development, and implementation. In this project, we follow the approach of a software ecosystem, that can be roughly explained as the software development and distribution by a set of actors dependent on each other and the ecosystem as a whole, and investigate the conditions of evolving a software ecosystem in the telemedicine services of Denmark. We do so by embarking on a set of quantitative and qualitative studies including literature surveys, case studies, and experiments.

This work contributes to the theory of software ecosystems by defining the concept and analysing the field while identifying areas of improvement. More-

over, we provide means of analyzing existing and designing new ecosystems, and defining and measuring the qualities of software ecosystems. Based on these contributions, we characterize the existing ecosystem in the telemedicine services of Denmark, identifying the lack of a common platform and actor and software interactions, findings that support our underlying hypothesis that the design of a software ecosystem would address the challenges of telemedicine. In continuation we design, implement, and evaluate Net4Care, a platform to facilitate the evolution of a software ecosystem in the telemedicine ecosystem. Furthermore, we provide input and guidance on how to initiate activities that would promote the establishment of this ecosystem through the 4S organization designed to take the role of an orchestrator in the ecosystem under development, managing the technological platform and promoting the actor interaction.

This work makes the initial steps towards a software ecosystem in the Danish telemedicine. An ecosystem that serves as means of arguably increasing the level of adoption of telemedicine technologies and eventually establishing telemedicine as state-wide method of patient diagnosis and treatment.

Appendix: Software ecosystem literature

1. Capuruço and Capretz (2010)
2. Popp (2011)
3. Yu and Deng (2011)
4. Molder et al. (2011)
5. Santos and Werner (2011b)
6. Angeren et al. (2011b)
7. Mens and Goeminne (2011)
8. Barbosa and Alves (2011)
9. Alspaugh et al. (2009)
10. Jansen et al. (2009a)
11. Fricker (2009)
12. Campbell and Ahmed (2010)
13. Seichter et al. (2010)
14. Dhungana et al. (2010)
15. Lungu et al. (2010b)
16. Berk et al. (2010)
17. Cataldo and Herbsleb (2010)
18. Goeminne and Mens (2010)
19. Bosch (2010a)
20. Bosch (2009)
21. Kazman and Chen (2010)
22. Lungu and Lanza (2010)
23. Pettersson et al. (2010)
24. Scacchi (2010b)
25. Boucharas et al. (2009)
26. Brummermann et al. (2011)
27. Anvaari and Jansen (2010)
28. Hunink et al. (2010)
29. Santos and Werner (2010)
30. McGregor (2010)
31. Scacchi (2007a)
32. Krishna and Srinivasa (2011)
33. Alves and Pessôa (2010)
34. Briscoe (2010)
35. Fricker (2010)
36. Scacchi (2010a)
37. Hilkert et al. (2010)
38. Bosch and Bosch-Sijtsema (2010c)
39. Bosch and Bosch-Sijtsema (2010a)
40. Kazman et al. (2012)
41. Schneider et al. (2010)
42. Yu and Woodard (2009)
43. Bosch (2010b)
44. Hanssen (2011)
45. Bosch and Bosch-Sijtsema (2010b)
46. Scacchi (2007b)
47. Lungu et al. (2010a)
48. Brewer and Johnson (2010)
49. Pettersson and Gil (2010)
50. Yu et al. (2008)
51. Lungu et al. (2009)
52. An (2009)
53. Janner et al. (2008)
54. Lungu (2008)
55. Hindle et al. (2010)
56. Jansen et al. (2009b)
57. Yu et al. (2007)
58. Schugerl et al. (2009)
59. Kakola (2010)
60. Ververs et al. (2011)

61. Angeren et al. (2011a)
62. Scholten et al. (2012)
63. Brummermann et al. (2012)
64. Stefanuto et al. (2011)
65. Schuur et al. (2011)
66. Ingen et al. (2011)
67. Weiss (2011)
68. Robbes and Lungu (2011)
69. Schmerl et al. (2011)
70. Yu (2011)
71. Santos and Werner (2011a)
72. Idu et al. (2011)
73. Draxler et al. (2011a)
74. Jergensen et al. (2011)
75. Scacchi and Alspaugh (2012)
76. Jansen et al. (2012)
77. Kilamo et al. (2012)
78. Pettersson and Vogel (2012)
79. Kajan et al. (2011)
80. Pérez et al. (2012)
81. Neu et al. (2011)
82. Riis and Schubert (2012)
83. Mizushima and Ikawa (2011)
84. Kabbedijk and Jansen (2011)
85. Draxler and Stevens (2011)
86. Burkard et al. (2012)
87. Viljainen and Kauppinen (2011)
88. Alves et al. (2011)
89. Draxler et al. (2011b)
90. Widjaja and Buxmann (2011)
91. Handoyo et al. (2013a)
92. Cardoso et al. (2013)
93. Costa et al. (2013)
94. Lettner et al. (2014)
95. Santos and Costa (2013)
96. Musil et al. (2013)
97. Seidl and Aßmann (2013)
98. Manikas and Hansen (2013a)
99. Lettner et al. (2013)
100. Seidl et al. (2013)
101. Keunecke et al. (2013)
102. Teixeira and Lin (2014)
103. Goeminne et al. (2013)
104. Howison and Herbsleb (2013)
105. Jansen (2013)
106. Monaco et al. (2013)
107. Musil et al. (2012)
108. Keivanloo (2012)
109. Satyanarayanan (2013)
110. Taylor (2013)
111. Santos (2014)
112. Haenni et al. (2013)
113. Mitropoulos et al. (2014)
114. Monteith et al. (2014)
115. Berger (2012)
116. Frantz and Corchuelo (2012)
117. Gunter et al. (2012)
118. Gutierrez and Robbes (2013)
119. Kästner et al. (2012)
120. Lopez (2013)

121. Linares-Vásquez et al. (2014)
122. Tymchuk et al. (2014)
123. Hoving et al. (2013)
124. Christensen and Hansen (2012)
125. German et al. (2013)
126. Yu (2013)
127. Bavota et al. (2013)
128. Dago (2012)
129. Kouters et al. (2012)
130. Fagerholm and Munch (2012)
131. Jaramillo et al. (2013)
132. Nöhren et al. (2014)
133. Santos and Werner (2012a)
134. Santos and Werner (2012b)
135. Keivanloo et al. (2012)
136. Claes et al. (2014)
137. Peniak et al. (2013)
138. Valenca (2013)
139. Bortolotti et al. (2013)
140. Schultis et al. (2013)
141. Tchoua et al. (2013)
142. Santos et al. (2012)
143. Pelliccione (2014)
144. Schmid (2013)
145. Mens et al. (2014a)
146. Amorim et al. (2014)
147. Schwarz et al. (2012)
148. McDonnell et al. (2013)
149. Hmood et al. (2012)
150. Goeminne (2014)
151. Andresen et al. (2013)
152. Che and Perry (2014)
153. Santos et al. (2013)
154. Albert et al. (2013)
155. Sasso and Lanza (2013)
156. Handoyo (2013)
157. Handoyo et al. (2013b)
158. Baars and Jansen (2012)
159. Hansen and Zhang (2014)
160. Rausch et al. (2013)
161. Wnuk et al. (2014a)
162. Eklund and Bosch (2012)
163. Wynn (2012)
164. Fotrousi et al. (2014)
165. Jansen and Bloemendal (2013)
166. Robbins and Tanik (2014)
167. Schütz et al. (2013)
168. Mens et al. (2014b)
169. Anvaari et al. (2013)
170. Waltl et al. (2012)
171. Bhowmik et al. (2014)
172. Vasilescu et al. (2014)
173. Kourtesis et al. (2012)
174. Bosch and Bosch-Sijtsema (2014)
175. Olsson and Bosch (2014)
176. Angeren et al. (2014)
177. Jansen (2014)
178. Manikas and Hansen (2013c)
179. Berger et al. (2014)
180. Dittrich (2014)

181. Bosch (2012)
182. Eklund and Bosch (2014)
183. Christensen et al. (2014)
184. Axelsson et al. (2014)
185. Wnuk et al. (2014b)
186. Hyrynsalmi et al. (2014)
187. Wright et al. (2013)
188. Iyer (2012)
189. Hanssen and Dybå (2012)
190. Salminen and Mikkonen (2012)
191. Popp (2012)
192. Jansen and Cusumano (2012)
193. Hyrynsalmi et al. (2012)
194. Santana and Werner (2013)
195. Syed and Jansen (2013)
196. Manikas and Hansen (2013b)
197. Lingen et al. (2013)
198. Monteith et al. (2013)
199. Spauwen and Jansen (2013)

References

Aanestad, M. and Jensen, T. B. (2011). Building nation-wide information infrastructures in healthcare through modular implementation strategies. *The Journal of Strategic Information Systems*, 20(2):161 – 176.

Ahgren, B. (2008). Is it better to be big?: The reconfiguration of 21st century hospitals: Responses to a hospital merger in sweden. *Health Policy*, 87(1):92–99.

Albert, B., Santos, R., and Werner, C. (2013). Software ecosystems governance to enable it architecture based on software asset management. In *Digital Ecosystems and Technologies (DEST), 2013 7th IEEE International Conference on*, pages 55–60.

Alspaugh, T., Asuncion, H., and Scacchi, W. (2009). The role of software licenses in open architecture ecosystems. In *First International Workshop on Software Ecosystems (IWSECO-2009)*, pages 4–18. Citeseer.

Alves, A. M. and Pessôa, M. (2010). Brazilian public software: beyond sharing. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '10, pages 73–80, New York, NY, USA. ACM.

Alves, A. M., Pessoa, M., and Salviano, C. F. (2011). Towards a systemic maturity model for public software ecosystems. In O'Connor, R. V., Rout, T., McCaffery, F., and Dorling, A., editors, *Software Process Improvement and Capability Determination*, volume 155 of *Communications in Computer and Information Science*, pages 145–156. Springer Berlin Heidelberg. 10.1007/978-3-642-21233-8_13.

Amorim, S. d. S., Almeida, E. S. d., and McGregor, J. D. (2014). Scalability of ecosystem architectures. In *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on*, pages 49–52.

An, H. (2009). Research on software problems based on ecological angle. In *Environmental Science and Information Application Technology, 2009. ESIAT 2009. International Conference on*, volume 3, pages 11 –14.

Andersen, P. and Jensen, J. (2010). Healthcare reform in denmark. *Scandinavian journal of public health*, 38(3):246–252.

Andresen, K., Brockmann, C., and Drager, C. (2013). A classification of ecosystems of enterprise system providers – an empirical analysis. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 4034–4044.

Anell, A., Glenngård, A. H., and Merkur, S. (2012). Sweeden. Health system review. *Health Systems in Transition*, 14(5):1–159.

Angeren, J. v., Blijleven, V., and Jansen, S. (2011a). Relationship intimacy in software ecosystems: a survey of the dutch software industry. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 68–75, New York, NY, USA. ACM.

Angeren, J. v., Jansen, S., and Brinkkemper, S. (2014). Exploring the relationship between partnership model participation and interfirm network structure: An analysis of the office365 ecosystem. In Lassenius, C. and Smolander, K., editors, *Software Business. Towards Continuous Value Delivery*, volume 182 of *Lecture Notes in Business Information Processing*, pages 1–15. Springer International Publishing.

Angeren, J. v., Kabbedijk, J., and Popp, K. M. (2011b). A survey of associate models used within large software ecosystems. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 27–39. CEUR-WS.

Anvaari, M., Conradi, R., and Jaccheri, L. (2013). Architectural decision-making in enterprises: Preliminary findings from an exploratory study in norwegian electricity industry. In Drira, K., editor, *Software Architecture*, volume 7957 of *Lecture Notes in Computer Science*, pages 162–175. Springer Berlin Heidelberg.

Anvaari, M. and Jansen, S. (2010). Evaluating architectural openness in mobile software platforms. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 85–92, New York, NY, USA. ACM.

Apple App Store (2013). Health & fitness - app store downloads on itunes. `https://itunes.apple.com/us/genre/ios-health-fitness/`, Accessed November 2013.

Axelsson, J., Papatheocharous, E., and Andersson, J. (2014). Characteristics of software ecosystems for federated embedded systems: A case study. *Information and Software Technology*, 56(11):1457 – 1475.

Baars, A. and Jansen, S. (2012). A framework for software ecosystem governance. In Cusumano, M., Iyer, B., and Venkatraman, N., editors, *Software Business*, volume 114 of *Lecture Notes in Business Information Processing*, pages 168–180. Springer Berlin Heidelberg.

Bamidis, P. D., Stamkopoulos, T.-G., Koufogiannis, D., Dombros, N., Maglaveras, N., and Pappas, C. (2006). Methodologies for establishing an institutional and regulatory framework for telemedicine services in greece. In *Proceedings of the 5th International IEEE EMBS Special Topic Conference on Information Technology in Biomedicin, Ioannina, Greece, 2006*.

Barbacci, M. R., Ellison, R., Lattanze, A. J., Stafford, J. A., Weinstock, C. B., and Wood, W. G. (2003). *Quality Attribute Workshops (QAWs)*. Software Engineering Institute, Carnegie Mellon University, third edition.

Barbosa, O. and Alves, C. (2011). A systematic mapping study on software ecosystems. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 15–26. CEUR-WS.

Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice*. Addison-Wesley, Boston, MA, USA, second edition.

Bavota, G., Canfora, G., Di Penta, M., Oliveto, R., and Panichella, S. (2013). The evolution of project inter-dependencies in a software ecosystem: The case of apache. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 280–289.

Berger, T. (2012). Variability modeling in the wild. In *Proceedings of the 16th International Software Product Line Conference - Volume 2*, SPLC '12, pages 233–241, New York, NY, USA. ACM.

Berger, T., Pfeiffer, R.-H., Tartler, R., Dienst, S., Czarnecki, K., Wąsowski, A., and She, S. (2014). Variability mechanisms in software ecosystems. *Information and Software Technology*, 56(11):1520 – 1535.

Berk, I. v. d., Jansen, S., and Luinenburg, L. (2010). Software ecosystems: a software ecosystem strategy assessment model. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 127–134, New York, NY, USA. ACM.

Bhanoo, S. N. (2010). Denmark leads the way in digital care. The New York Times.

Bhowmik, T., Alves, V., and Niu, N. (2014). An exploratory case study on exploiting aspect orientation in mobile game porting. In Bouabana-Tebibel, T. and Rubin, S. H., editors, *Integration of Reusable Systems*, volume 263 of *Advances in Intelligent Systems and Computing*, pages 241–261. Springer International Publishing.

Boone, K. W. (2011). *The CDA Book*. Springer.

Bortolotti, D., Pinto, C., Marongiu, A., Ruggiero, M., and Benini, L. (2013). Virtualsoc: A full-system simulation environment for massively parallel heterogeneous system-on-chip. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 2182–2187.

Bosch, J. (2009). From software product lines to software ecosystems. In *Proceedings of the 13th International Software Product Line Conference*, SPLC '09, pages 111–119, Pittsburgh, PA, USA. Carnegie Mellon University.

Bosch, J. (2010a). Architecture challenges for software ecosystems. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 93–95, New York, NY, USA. ACM.

Bosch, J. (2010b). Architecture in the age of compositionality. In Babar, M. and Gorton, I., editors, *Software Architecture*, volume 6285 of *Lecture Notes in Computer Science*, pages 1–4. Springer Berlin / Heidelberg. 10.1007/978-3-642-15114-9_1.

Bosch, J. (2012). Software ecosystems: Taking software development beyond the boundaries of the organization. *Journal of Systems and Software*, 85(7):1453 – 1454.

Bosch, J. and Bosch-Sijtsema, P. (2010a). Coordination between global agile teams: From process to architecture. In Ŝmite, D., Moe, N. B., and Agerfalk, P. J., editors, *Agility Across Time and Space*, pages 217–233. Springer Berlin Heidelberg. 10.1007/978-3-642-12442-6_15.

Bosch, J. and Bosch-Sijtsema, P. (2010b). From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1):67 – 76.

Bosch, J. and Bosch-Sijtsema, P. (2014). Esao: A holistic ecosystem-driven analysis model. In Lassenius, C. and Smolander, K., editors, *Software Business. Towards Continuous Value Delivery*, volume 182 of *Lecture Notes in Business Information Processing*, pages 179–193. Springer International Publishing.

Bosch, J. and Bosch-Sijtsema, P. M. (2010c). Softwares product lines, global development and ecosystems: Collaboration in software engineering. In Mistrík, I., van der Hoek, A., Grundy, J., and Whitehead, J., editors, *Collaborative Software Engineering*, pages 77–92. Springer Berlin Heidelberg. 10.1007/978-3-642-10294-3_4.

Boucharas, V., Jansen, S., and Brinkkemper, S. (2009). Formalizing software ecosystem modeling. In *Proceedings of the 1st international workshop on Open component ecosystems*, IWOCE '09, pages 41–50, New York, NY, USA. ACM.

Brewer, R. and Johnson, P. (2010). Wattdepot: An open source software ecosystem for enterprise-scale energy data collection, storage, analysis, and visualization. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 91 –95.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

Briscoe, G. (2010). Complex adaptive digital ecosystems. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '10, pages 39–46, New York, NY, USA. ACM.

Brummermann, H., Keunecke, M., and Schmid, K. (2011). Variability issues in the evolution of information system ecosystems. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '11, pages 159–164, New York, NY, USA. ACM.

Brummermann, H., Keunecke, M., and Schmid, K. (2012). Formalizing distributed evolution of variability in information system ecosystems. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '12, pages 11–19, New York, NY, USA. ACM.

Burkard, C., Widjaja, T., and Buxmann, P. (2012). Software ecosystems. *Business & Information Systems Engineering*, 4:41–44. 10.1007/s12599-011-0199-8.

Campbell, D. T., Stanley, J. C., and Gage, N. L. (1963). *Experimental and quasi-experimental designs for research*. Houghton Mifflin Boston.

Campbell, P. R. J. and Ahmed, F. (2010). A three-dimensional view of software ecosystems. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 81–84, New York, NY, USA. ACM.

Capuruço, R. A. C. and Capretz, L. F. (2010). Integrating recommender information in social ecosystems decisions. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 143–150, New York, NY, USA. ACM.

Cardoso, Jr., J. L., Barbin, S. E., Andres, F., and Filho, O. S. S. (2013). The public software ecosystem: Exploratory survey. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems*, MEDES '13, pages 289–296, New York, NY, USA. ACM.

Cataldo, M. and Herbsleb, J. D. (2010). Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 65–72, New York, NY, USA. ACM.

Che, M. and Perry, D. E. (2014). Architectural design decisions in open software development: A transition to software ecosystems. In *Software Engineering Conference (ASWEC), 2014 23rd Australian*, pages 58–61.

Chevreul, K., Durand-Zaleski, I., Bahrami, S., Hernández-Quevedo, C., and Mladovsky, P. (2010). France. Health system review. *Health Systems in Transition*, 12(6):1–291.

Christensen, A. I., Davidsen, M., Ekholm, O., Hansen, S. E., Holst, M., and Jue, K. (2011). *Den nationale sundhedsprofil 2010 – hvordan har du det?* Sundhedsstyrelsen.

Christensen, H. and Hansen, K. (2012). Net4care: Towards a mission-critical software ecosystem. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, pages 224–228.

Christensen, H. B., Hansen, K. M., Kyng, M., and Manikas, K. (2014). Analysis and design of software ecosystem architectures – towards the 4s telemedicine ecosystem. *Information and Software Technology*, 56(11):1476 – 1492.

Claes, M., Mens, T., and Grosjean, P. (2014). On the maintainability of cran packages. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 308–312.

Cohen, M. D. and Hilligoss, P. B. (2010). The published literature on handoffs in hospitals: deficiencies identified in an extensive review. *Quality and Safety in Health Care*.

Costa, G., Silva, F., Santos, R., Werner, C., and Oliveira, T. (2013). From applications to a software ecosystem platform: An exploratory study. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems*, MEDES '13, pages 9–16, New York, NY, USA. ACM.

Cusumano, M. A. (2004). *The business of software: What every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad*. Free Press.

Dago, G. (2012). Creating an software ecosystem in order to preserve the first argentine computer language and compiler: A case study on computing archaeology. In *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, pages 1–9.

Danish Ministry of Health and Prevention (2008). Health care in denmark.

Danmark Statistik (2011). NYT fra Danmarks Statistik – Befolkningsfremskrivninger 2011-2050. `http://www.dst.dk/nytudg/14607`. Accessed July 2014.

Danske Regioner (2007). Investeringer i fremtidens sundhedsvæsen.

Delloitte (2007). Bestyrelsen for den nationale epj-organisation, strategiske udviklingsveje for epj.

den Hartigh, E., Tol, M., and Visscher, W. (2006). The health measurement of a business ecosystem. In *Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting*, page 0.

Denscombe, M. (2010). *The good research guide – for small-scale social research projects*. Open University Press, second edition.

Denzin, N. (1978). *The Research Act: A Theoretical Introduction to Sociological Methods*. McGraw-Hill.

Dhungana, D., Groher, I., Schludermann, E., and Biffl, S. (2010). Software ecosystems vs. natural ecosystems: learning from the ingenious mind of nature. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 96–102, New York, NY, USA. ACM.

Dittrich, Y. (2014). Software engineering beyond the project – sustaining software ecosystems. *Information and Software Technology*, 56(11):1436 – 1456.

Draxler, S., Jung, A., Boden, A., and Stevens, G. (2011a). Workplace warriors: identifying team practices of appropriation in software ecosystems. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '11, pages 57–60, New York, NY, USA. ACM.

Draxler, S., Jung, A., and Stevens, G. (2011b). Managing software portfolios: A comparative study. In Costabile, M., Dittrich, Y., Fischer, G., and Piccinno, A., editors, *End-User Development*, volume 6654 of *Lecture Notes in Computer Science*, pages 337–342. Springer Berlin / Heidelberg. 10.1007/978-3-642-21530-8_36.

Draxler, S. and Stevens, G. (2011). Supporting the collaborative appropriation of an open software ecosystem. *Computer Supported Cooperative Work (CSCW)*, 20:403–448. 10.1007/s10606-011-9148-9.

Economou, C. (2010). Greece. Health system review. *Health Systems in Transition*, 12(7):1–180.

EgenJournal (2010). CITH Co-constructing IT and Healthcare. `http://www.cith.dk/`.

Eklund, U. and Bosch, J. (2012). Introducing software ecosystems for mass-produced embedded systems. In Cusumano, M., Iyer, B., and Venkatraman, N., editors, *Software Business*, volume 114 of *Lecture Notes in Business Information Processing*, pages 248–254. Springer Berlin Heidelberg.

Eklund, U. and Bosch, J. (2014). Architecture for embedded open software ecosystems. *Journal of Systems and Software*, 92(0):128 – 142.

Esping-Andersen, G. (1990). *The three worlds of welfare capitalism*, volume 6. Polity press Cambridge.

Fagerholm, F. and Munch, J. (2012). Developer experience: Concept and definition. In *Software and System Process (ICSSP), 2012 International Conference on*, pages 73–77.

Fotrousi, F., Fricker, S., Fiedler, M., and Le-Gall, F. (2014). Kpis for software ecosystems: A systematic mapping study. In Lassenius, C. and Smolander, K., editors, *Software Business. Towards Continuous Value Delivery*, volume 182 of *Lecture Notes in Business Information Processing*, pages 194–211. Springer International Publishing.

Francesca, C., Ana, L.-N., Jérôme, M., and Frits, T. (2011). *OECD Health Policy Studies Help Wanted? Providing and Paying for Long-Term Care*, volume 2011. OECD Publishing.

Frantz, R. Z. and Corchuelo, R. (2012). A software development kit to implement integration solutions. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1647–1652, New York, NY, USA. ACM.

Fricker, S. (2009). Specification and analysis of requirements negotiation strategy in software ecosystems. In *First International Workshop on Software Ecosystems (IWSECO-2009)*, pages 19–33. Citeseer.

Fricker, S. (2010). Requirements value chains: Stakeholder management and requirements engineering in software ecosystems. In Wieringa, R. and Persson, A., editors, *Requirements Engineering: Foundation for Software Quality*, volume 6182 of *Lecture Notes in Computer Science*, pages 60–66. Springer Berlin / Heidelberg. 10.1007/978-3-642-14192-8_7.

Fulop, N., Protopsaltis, G., Hutchings, A., King, A., Allen, P., Normand, C., and Walters, R. (2002). Process and impact of mergers of NHS trusts: multicentre case study and management cost analysis. *BMJ*, 325(7358):246.

German, D., Adams, B., and Hassan, A. (2013). The evolution of the r software ecosystem. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, pages 243–252.

Goeminne, M. (2014). Understanding the evolution of socio-technical aspects in open source ecosystems. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 473–476.

Goeminne, M., Claes, M., and Mens, T. (2013). A historical dataset for the gnome ecosystem. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 225–228, Piscataway, NJ, USA. IEEE Press.

Goeminne, M. and Mens, T. (2010). A framework for analysing and visualising open source software ecosystems. In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, IWPSE-EVOL '10, pages 42–47, New York, NY, USA. ACM.

Gunter, D., Cholia, S., Jain, A., Kocher, M., Persson, K., Ramakrishnan, L., Ong, S. P., and Ceder, G. (2012). Community accessible datastore of high-throughput calculations: Experiences from the materials project. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 1244–1251.

Gutierrez, C. and Robbes, R. (2013). Weon: Towards a software ecosystem ontology. In *Proceedings of the 2013 International Workshop on Ecosystem Architectures*, WEA 2013, pages 16–20, New York, NY, USA. ACM.

Haenni, N., Lungu, M., Schwarz, N., and Nierstrasz, O. (2013). Categorizing developer information needs in software ecosystems. In *Proceedings of the 2013 International Workshop on Ecosystem Architectures*, WEA 2013, pages 1–5, New York, NY, USA. ACM.

Handoyo, E. (2013). Software ecosystem modeling. In Herzwurm, G. and Margaria, T., editors, *Software Business. From Physical Products to Software Services and Solutions*, volume 150 of *Lecture Notes in Business Information Processing*, pages 227–228. Springer Berlin Heidelberg.

Handoyo, E., Jansen, S., and Brinkkemper, S. (2013a). Software ecosystem modeling: The value chains. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems*, MEDES '13, pages 17–24, New York, NY, USA. ACM.

Handoyo, E., Jansen, S., and Brinkkemper, S. (2013b). Software ecosystem roles classification. In Herzwurm, G. and Margaria, T., editors, *Software Business. From Physical Products to Software Services and Solutions*, volume 150 of *Lecture Notes in Business Information Processing*, pages 212–216. Springer Berlin Heidelberg.

Hansen, K. and Zhang, W. (2014). Towards structure-based quality awareness in software ecosystem use. In Lomuscio, A., Nepal, S., Patrizi, F., Benatallah, B., and Brandić, I., editors, *Service-Oriented Computing - ICSOC 2013 Workshops*, volume 8377 of *Lecture Notes in Computer Science*, pages 469–479. Springer International Publishing.

Hansen, K. M. and Manikas, K. (2013). Towards a Network Ecology of Software Ecosystems: an Analysis of two OSGi Ecosystems. In *Proceedings of the 25th International Conference on Software Engineering & Knowledge Engineering (SEKE'2013)*, pages 326–331.

Hanssen, G. K. (2011). A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. *Journal of Systems and Software*, 85(7):1455 – 1466.

Hanssen, G. K. and Dybå, T. (2012). Theoretical foundations of software ecosystems. In Jansen, S., Bosch, J., and Alves, C., editors, *Proceedings of the Forth International Workshop on Software Ecosystems, Cambridge, MA, USA, June 18th, 2012*, volume 879, pages 6–17. CEUR-WS.org.

Hilkert, D., Wolf, C. M., Benlian, A., and Hess, T. (2010). The "as-a-service"-paradigm and its implications for the software industry – insights from a comparative case study in crm software ecosystems. In Aalst, W., Mylopoulos, J., Sadeh, N. M., Shaw, M. J., Szyperski, C., Tyrvainen, P., Jansen, S., and Cusumano, M. A., editors, *Software Business*, volume 51 of *Lecture Notes in Business Information Processing*, pages 125–137. Springer Berlin Heidelberg. 10.1007/978-3-642-13633-7_11.

Hindle, A., Herraiz, I., Shihab, E., and Jiang, Z. M. (2010). Mining challenge 2010: Freebsd, gnome desktop and debian/ubuntu. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 82 –85.

HL7 (2010). Implementation Guide for CDA Release 2.0 Personal Healthcare Monitoring Report (PHMR) (International Realm) Draft Standard for Trial Use Release 1.1.

HL7 (2014). Hl7: Health level seven international. `http://www.hl7.org`. Accessed April 2014.

Hmood, A., Keivanloo, I., and Rilling, J. (2012). Se-equam - an evolvable quality metamodel. In *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, pages 334–339.

Hoving, R., Slot, G., and Jansen, S. (2013). Python: Characteristics identification of a free open source software ecosystem. In *Digital Ecosystems and Technologies (DEST), 2013 7th IEEE International Conference on*, pages 13–18.

Howison, J. and Herbsleb, J. D. (2013). Incentives and integration in scientific software production. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 459–470, New York, NY, USA. ACM.

Hunink, I., van Erk, R., Jansen, S., and Brinkkemper, S. (2010). Industry taxonomy engineering: the case of the european software ecosystem. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 111–118, New York, NY, USA. ACM.

Hyrynsalmi, S., Mäkilä, T., Järvi, A., Suominen, A., Seppänen, M., and Knuutila, T. (2012). App store, marketplace, play! an analysis of multi-homing in mobile software ecosystems. In Jansen, S., Bosch, J., and Alves, C., editors, *Proceedings of the Forth International Workshop on Software Ecosystems, Cambridge, MA, USA, June 18th, 2012*, volume 879, pages 59 – 72. CEUR-WS.org.

Hyrynsalmi, S., Seppänen, M., and Suominen, A. (2014). Sources of value in application ecosystems. *Journal of Systems and Software*, 96(0):61 – 72.

Iansiti, M. and Levien, R. (2004a). *The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability*. Harvard Business Press.

Iansiti, M. and Levien, R. (2004b). Keystones and dominators: Framing operating and technology strategy in a business ecosystem. *Harvard Business School, Boston*.

Iansiti, M. and Levien, R. (2004c). Strategy as ecology. *Harvard Business Review*, 82(3):68–81.

Iansiti, M. and Richards, G. L. (2006). The information technology ecosystem: Structure, health, and performance. *Antitrust Bull.*, 51:77.

Idu, A., van de Zande, T., and Jansen, S. (2011). Multi-homing in the apple ecosystem: why and how developers target multiple apple app stores. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 122–128, New York, NY, USA. ACM.

IHE (2013). IT Infrastructure Technical Framework. Volume 1 (ITI TF-1). Integration Profiles. Revision 9.0.

Ingen, K. v., van Ommen, J., and Jansen, S. (2011). Improving activity in communities of practice through software release management. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 94–98, New York, NY, USA. ACM.

IWSECO (2014). IWSECO | International Workshop on Software Ecosystems. `http://iwseco.org/` Accessed August, 2014.

Iyer, B. (2012). Invited paper: Ecosysnetworks: A method for visualizing software ecosystems. In Jansen, S., Bosch, J., and Alves, C., editors, *Proceedings of the Forth International Workshop on Software Ecosystems, Cambridge, MA, USA, June 18th, 2012*, volume 879, pages 1–5. CEUR-WS.org.

Janner, T., Schroth, C., and Schmid, B. (2008). Modelling service systems for collaborative innovation in the enterprise software industry - the st. gallen media reference model applied. In *Services Computing, 2008. SCC '08. IEEE International Conference on*, volume 2, pages 145 –152.

Jansen, S. (2013). How quality attributes of software platform architectures influence software ecosystems. In *Proceedings of the 2013 International Workshop on Ecosystem Architectures*, WEA 2013, pages 6–10, New York, NY, USA. ACM.

Jansen, S. (2014). Measuring the health of open source software ecosystems: Beyond the scope of project health. *Information and Software Technology*, 56(11):1508 – 1519.

Jansen, S. and Bloemendal, E. (2013). Defining app stores: The role of curated marketplaces in software ecosystems. In Herzwurm, G. and Margaria, T., editors, *Software Business. From Physical Products to Software Services and Solutions*, volume 150 of *Lecture Notes in Business Information Processing*, pages 195–206. Springer Berlin Heidelberg.

Jansen, S., Brinkkemper, S., and Finkelstein, A. (2009a). Business network management as a survival strategy: A tale of two software ecosystems. In *First International Workshop on Software Ecosystems (IWSECO-2009)*, pages 34–48. Citeseer.

Jansen, S., Brinkkemper, S., Souer, J., and Luinenburg, L. (2012). Shades of gray: Opening up a software producing organization with the open software enterprise model. *Journal of Systems and Software*, 85(7):1495 – 1510.

Jansen, S. and Cusumano, M. (2012). Defining software ecosystems: A survey of software platforms and business network governance. In Jansen, S., Bosch, J., and Alves, C., editors, *Proceedings of the Forth International Workshop on Software Ecosystems, Cambridge, MA, USA, June 18th, 2012*, volume 879, pages 40 – 58. CEUR-WS.org.

Jansen, S. and Cusumano, M. (2013). Software ecosystems – analyzing and managing business networks in the software industry. In Jansen, S., Brinkkemper, S., and Cusumano, M., editors, *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, Cheltenham, UK.

Jansen, S., Finkelstein, A., and Brinkkemper, S. (2009b). A sense of community: A research agenda for software ecosystems. In *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 187 –190.

Jaramillo, D., Newhook, R., and Smart, R. (2013). Cross-platform, secure message delivery for mobile devices. In *Southeastcon, 2013 Proceedings of IEEE*, pages 1–5.

Jergensen, C., Sarma, A., and Wagstrom, P. (2011). The onion patch: migration in open source ecosystems. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, ESEC/FSE '11, pages 70–80, New York, NY, USA. ACM.

Jordán, F. and Scheuring, I. (2004). Network ecology: topological constraints on ecosystem dynamics. *Physics of Life Reviews*, 1(3):139–172.

Kabbedijk, J. and Jansen, S. (2011). Steering insight: An exploration of the ruby software ecosystem. In Regnell, B., Weerd, I., Troyer, O., Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M. J., and Szyperski, C., editors, *Software Business*, volume 80 of *Lecture Notes in Business Information Processing*, pages 44–55. Springer Berlin Heidelberg. 10.1007/978-3-642-21544-5_5.

Kajan, E., Lazic, L., and Maamar, Z. (2011). Software engineering framework for digital service-oriented ecosystem. In *Telecommunications Forum (TELFOR), 2011 19th*, pages 1320 –1323.

Kakola, T. (2010). Standards initiatives for software product line engineering and management within the international organization for standardization. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1 –10.

Kästner, C., Ostermann, K., and Erdweg, S. (2012). A variability-aware module system. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 773–792, New York, NY, USA. ACM.

Kazman, R. and Chen, H.-M. (2010). The metropolis model and its implications for the engineering of software ecosystems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 187–190, New York, NY, USA. ACM.

Kazman, R., Gagliardi, M., and Wood, W. (2012). Scaling up software architecture analysis. *Journal of Systems and Software*, 85(7):1511 – 1519.

Keivanloo, I. (2012). Online sharing and integration of results from mining software repositories. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 1644–1646, Piscataway, NJ, USA. IEEE Press.

Keivanloo, I., Forbes, C., Hmood, A., Erfani, M., Neal, C., Peristerakis, G., and Rilling, J. (2012). A linked data platform for mining software repositories. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 32–35.

Keunecke, M., Brummermann, H., and Schmid, K. (2013). The feature pack approach: Systematically managing implementations in software ecosystems. In *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, VaMoS '14, pages 20:1–20:7, New York, NY, USA. ACM.

Kierkegaard, P. (2013). ehealth in denmark: A case study. *Journal of medical systems*, 37(6):1–10.

Kilamo, T., Hammouda, I., Mikkonen, T., and Aaltonen, T. (2012). From proprietary to open source–growing an open source ecosystem. *Journal of Systems and Software*, 85(7):1467 – 1478.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. *Engineering*, 2(EBSE 2007-001).

Kourtesis, D., Bratanis, K., Bibikas, D., and Paraskakis, I. (2012). Software co-development in the era of cloud application platforms and ecosystems: The case of cast. In Camarinha-Matos, L., Xu, L., and Afsarmanesh, H., editors, *Collaborative Networks in the Internet of Services*, volume 380 of *IFIP Advances in Information and Communication Technology*, pages 196–204. Springer Berlin Heidelberg.

Kouters, E., Vasilescu, B., Serebrenik, A., and van den Brand, M. (2012). Who's who in gnome: Using lsa to merge software repository identities. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 592–595.

Krishna, R. P. M. and Srinivasa, K. G. (2011). Analysis of projects and volunteer participation in large scale free and open source software ecosystem. *SIGSOFT Softw. Eng. Notes*, 36:1–5.

Kristensen, T., Olsen, K. R., Kilsmark, J., and Pedersen, K. M. (2008). Economies of scale and optimal size of hospitals: Empirical results for danish public hospitals. Workingpaper, Syddansk Universitet.

Lettner, D., Angerer, F., Prähofer, H., and Grünbacher, P. (2014). A case study on software ecosystem characteristics in industrial automation software. In *Proceedings of the 2014 International Conference on Software and System Process*, ICSSP 2014, pages 40–49, New York, NY, USA. ACM.

Lettner, D., Petruzelka, M., Rabiser, R., Angerer, F., Prähofer, H., and Grünbacher, P. (2013). Custom-developed vs. model-based configuration tools: Experiences from an industrial automation ecosystem. In *Proceedings of the 17th International Software Product Line Conference Co-located Workshops*, SPLC '13 Workshops, pages 52–58, New York, NY, USA. ACM.

Linares-Vásquez, M., Bavota, G., Di Penta, M., Oliveto, R., and Poshyvanyk, D. (2014). How do api changes trigger stack overflow discussions? a study on the android sdk. In *Proceedings of the 22Nd International Conference on Program Comprehension*, ICPC 2014, pages 83–94, New York, NY, USA. ACM.

Lingen, S. v., Palomba, A., and Lucassen, G. (2013). On the software ecosystem health of open source content management systems. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 45–56. CEUR-WS.org.

Lopez, N. (2013). Using topic models to understand the evolution of a software ecosystem. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 723–726, New York, NY, USA. ACM.

Lund, S. R. (2013). Staten ramt af ny it-fadæse - kommunen. `http://kommunen.dk/Artikel/?id=2997`. Accessed June 2014.

Lungu, M. (2008). Towards reverse engineering software ecosystems. In *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, pages 428 –431.

Lungu, M. and Lanza, M. (2010). The small project observatory: a tool for reverse engineering software ecosystems. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pages 289–292, New York, NY, USA. ACM.

Lungu, M., Lanza, M., Gîrba, T., and Robbes, R. (2010a). The small project observatory: Visualizing software ecosystems. *Science of Computer Programming*, 75(4):264 – 275. Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008).

Lungu, M., Malnati, J., and Lanza, M. (2009). Visualizing gnome with the small project observatory. In *Mining Software Repositories, 2009. MSR '09. 6th IEEE International Working Conference on*, pages 103 –106.

Lungu, M., Robbes, R., and Lanza, M. (2010b). Recovering inter-project dependencies in software ecosystems. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ASE '10, pages 309–312, New York, NY, USA. ACM.

Magnussen, J., Hagen, T. P., and Kaarboe, O. M. (2007). Centralized or decentralized? a case study of norwegian hospital reform. *Social science & medicine*, 64(10):2129–2137.

Manikas, K. and Hansen, K. M. (2013a). Characterizing the danish telemedicine ecosystem: Making sense of actor relationships. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems*, MEDES '13, pages 211–218.

Manikas, K. and Hansen, K. M. (2013b). Reviewing the health of software ecosystems - a conceptual framework proposal. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 33–44.

Manikas, K. and Hansen, K. M. (2013c). Software ecosystems – a systematic literature review. *Journal of Systems and Software*, 86(5):1294 – 1306.

Manikas, K., Hansen, K. M., and Kyng, M. (2014). Governance mechanisms for healthcare apps. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 10:1–10:6, New York, NY, USA. ACM.

McDonnell, T., Ray, B., and Kim, M. (2013). An empirical study of api stability and adoption in the android ecosystem. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 70–79.

McGregor, J. D. (2010). A method for analyzing software product line ecosystems. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 73–80, New York, NY, USA. ACM.

MedCom (2013). Overview of danish telemedical initiatives. `https://medcom.medware.dk/telemedicine_projects`. Accessed December 2013.

MedCom (2014a). Sundhedsdatanettet (SDN). `http://www.medcom.dk/wm110002`. Accessed July 2014.

MedCom (2014b). Videokonference (VDX). `http://www.medcom.dk/wm110004`. Accessed July 2014.

Mens, T., Claes, M., and Grosjean, P. (2014a). Ecos: Ecological studies of open source software ecosystems. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 403–406.

Mens, T., Claes, M., Grosjean, P., and Serebrenik, A. (2014b). Studying evolving software ecosystems based on ecological models. In Mens, T., Serebrenik, A., and Cleve, A., editors, *Evolving Software Systems*, pages 297–326. Springer Berlin Heidelberg.

Mens, T. and Goeminne, M. (2011). Analysing the evolution of social aspects of open source software ecosystems. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 1–14. CEUR-WS.

Messerschmitt, D. and Szyperski, C. (2003). Software ecosystem: understanding an indispensable technology and industry. *MIT Press Books*, 1.

Mitropoulos, D., Karakoidas, V., Louridas, P., Gousios, G., and Spinellis, D. (2014). The bug catalog of the maven ecosystem. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 372–375, New York, NY, USA. ACM.

Mizushima, K. and Ikawa, Y. (2011). A structure of co-creation in an open source software ecosystem: A case study of the eclipse community. In *Technology Management in the Energy Smart World (PICMET), 2011 Proceedings of PICMET '11:*, pages 1 –8.

Molder, J., van Lier, B., and Jansen, S. (2011). Clopenness of systems: The interwoven nature of ecosystems. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 52–64. CEUR-WS.

Monaco, M., Michel, O., and Keller, E. (2013). Applying operating system principles to sdn controller design. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, pages 2:1–2:7, New York, NY, USA. ACM.

Monteith, J. Y., McGregor, J. D., and Ingram, J. E. (2013). Hadoop and its evolving ecosystem. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 57–68. CEUR-WS.org.

Monteith, J. Y., McGregor, J. D., and Ingram, J. E. (2014). Proposed metrics on ecosystem health. In *Proceedings of the 2014 ACM International Workshop on Software-defined Ecosystems*, BigSystem '14, pages 33–36, New York, NY, USA. ACM.

Musil, J., Musil, A., and Biffl, S. (2013). Elements of software ecosystem early-stage design for collective intelligence systems. In *Proceedings of the 2013 International Workshop on Ecosystem Architectures*, WEA 2013, pages 21–25, New York, NY, USA. ACM.

Musil, J., Musil, A., Winkler, D., and Biffl, S. (2012). A first account on stigmergic information systems and their impact on platform development. In *Proceedings of the WICSA/ECSA 2012 Companion Volume*, WICSA/ECSA '12, pages 69–73, New York, NY, USA. ACM.

Mylonas, A., Kastania, A., and Gritzalis, D. (2013). Delegate the smartphone user? security awareness in smartphone platforms. *Computers & Security*, 34:47–66.

Net4Care (2014). Net4care - the net4care ecosystem platform. `http://net4care.org/` Accessed August, 2014.

Neu, S., Lanza, M., Hattori, L., and D'Ambros, M. (2011). Telling stories about gnome with complicity. In *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*, pages 1 –8.

Null, C. (2013). The end of symbian: Nokia ships last handset with the mobile os. `http://www.pcworld.com/article/2042071/the-end-of-symbian-nokia-ships-last-handset-with-the-mobile-os.html`. Accessed September 2014.

Nöhren, M., Heinzl, A., and Kude, T. (2014). Structural and behavioral fit in software sourcing alignment. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 3949–3958.

OECD (2014). Health policies and data – OECD Health Statistics 2014. `http://www.oecd.org/els/health-systems/health-data.htm`. Accessed Jully 2014.

Olejaz, M., Nielsen, A. J., Rudkjøbing, A., Birk, H. O., Krasnik, A., and Hernández-Quevedo, C. (2012). Denmark. Health system review. *Health Systems in Transition*, 14(2):1–192.

Olsson, H. and Bosch, J. (2014). Ecosystem-driven software development: A case study on the emerging challenges in inter-organizational r&d. In Lassenius, C. and Smolander, K., editors, *Software Business. Towards Continuous Value Delivery*, volume 182 of *Lecture Notes in Business Information Processing*, pages 16–26. Springer International Publishing.

Orthacker, C., Teufl, P., Kraxberger, S., Lackner, G., Gissing, M., Marsalek, A., Leibetseder, J., and Prevenhueber, O. (2012). Android security permissions – can we trust them? In Prasad, R., Farkas, K., Schmidt, A., Lioy, A., Russello, G., and Luccio, F., editors, *Security and Privacy in Mobile Information and Communication Systems*, volume 94 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 40–51. Springer Berlin Heidelberg.

Patton, M. Q. (2002). *Qualitative research and evaluation methods*. SAGE Publications, inc.

Pelliccione, P. (2014). Open architectures and software evolution: The case of software ecosystems. In *Software Engineering Conference (ASWEC), 2014 23rd Australian*, pages 66–69.

Peniak, M., Morse, A., and Cangelosi, A. (2013). Aquila 2.0 software architecture for cognitive robotics. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, pages 1–6.

Pérez, J., Deshayes, R., Goeminne, M., and Mens, T. (2012). Seconda: Software ecosystem analysis dashboard. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pages 527 –530.

Pettersson, O. and Gil, D. (2010). On the issue of reusability and adaptability in m-learning systems. In *Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE), 2010 6th IEEE International Conference on*, pages 161 –165.

Pettersson, O., Svensson, M., Gil, D., Andersson, J., and Milrad, M. (2010). On the role of software process modeling in software ecosystem design. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 103–110, New York, NY, USA. ACM.

Pettersson, O. and Vogel, B. (2012). Reusability and interoperability in mobile learning: A study of current practices. In *Wireless, Mobile and Ubiquitous Technology in Education (WMUTE), 2012 IEEE Seventh International Conference on*, pages 306 –310.

Popp, K. M. (2011). Hybrid revenue models of software companies and their relationship to hybrid business models. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 77–88. CEUR-WS.

Popp, K. M. (2012). Leveraging open source licenses and open source communities in hybrid commercial open source business models. In Jansen, S., Bosch, J., and Alves, C., editors, *Proceedings of the Forth International Workshop on Software Ecosystems, Cambridge, MA, USA, June 18th, 2012*, volume 879, pages 33 – 40. CEUR-WS.org.

Protti, D. and Johansen, I. (2010). Widespread adoption of information technology in primary care physician offices in denmark: a case study. *Issue brief (Commonwealth Fund)*, 80:1–14.

Quantified Self (2014). Quantified self – self knowledge through numbers. `http://quantifiedself.com/`, Accessed July 2014.

Rausch, A., Bartelt, C., Herold, S., Klus, H., and Niebuhr, D. (2013). From software systems to complex software ecosystems: Model- and constraint-based engineering of ecosystems. In Münch, J. and Schmid, K., editors, *Perspectives on the Future of Software Engineering*, pages 61–80. Springer Berlin Heidelberg.

Riis, P. and Schubert, P. (2012). Upgrading to a new version of an erp system: A multilevel analysis of influencing factors in a software ecosystem. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 4709 –4718.

Ringard, Å., Sagan, A., Saunes, I. S., and Lindahl, A. K. (2013). Norway. Health system review. *Health Systems in Transition*, 15(8):1–162.

Robbes, R. and Lungu, M. (2011). A study of ripple effects in software ecosystems (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 904–907, New York, NY, USA. ACM.

Robbins, D. and Tanik, M. (2014). Cyber-physical ecosystems: App-centric software ecosystems in cyber-physical environments. In Suh, S. C., Tanik, U. J., Carbone, J. N., and Eroglu, A., editors, *Applied Cyber-Physical Systems*, pages 141–147. Springer New York.

Robson, C. (2011). *Real world research.* Wiley, third edition.

Rodrigues, R., Huber, M., and Lamura, G., editors (2012). *Facts and Figures on Healthy Ageing and Long-term Care – Europe and North America.* European Centre for Social Welfare Policy and Research.

RRS (2009). Remote rehabilitation support. `http://www.caretechinnovation.dk/en/projects/rrs.htm`.

Rudkjøbing, A., Olejaz, M., Birk, H. O., Nielsen, A. J., Hernández-Quevedo, C., and Krasnik, A. (2012). Integrated care: a danish perspective. *BMJ*, 345(2012):4451–4451.

Runeson, P., Höst, M., Rainer, A., and Regnell, B. (2012). *Case Study Research in Software Engineering – Guidelines and Examples.* Wiley.

Salminen, A. and Mikkonen, T. (2012). Mashups - software ecosystems for the web era. In Jansen, S., Bosch, J., and Alves, C., editors, *Proceedings of the Forth International Workshop on Software Ecosystems, Cambridge, MA, USA, June 18th, 2012*, volume 879, pages 18 – 32. CEUR-WS.org.

Santana, F. W. and Werner, C. M. L. (2013). Towards the analysis of software projects dependencies: An exploratory visual study of software ecosystems. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 7–18. CEUR-WS.org.

Santos, E. and Costa, L. F. (2013). Brazil and south africa collaboration for public software: Building the south africa public software ecosystem. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems*, MEDES '13, pages 314–319, New York, NY, USA. ACM.

Santos, P. R., Tostes, R. L., and Werner, L. C. (2013). A brechó-ecosys extension to support negotiation in the software ecosystems context. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 578–585.

Santos, R. and Werner, C. (2012a). Reuseecos: An approach to support global software development through software ecosystems. In *Global Software Engineering Workshops (ICGSEW), 2012 IEEE Seventh International Conference on*, pages 60–65.

Santos, R. and Werner, C. (2012b). Treating social dimension in software ecosystems through reuseecos approach. In *Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on*, pages 1–6.

Santos, R., Werner, C., Barbosa, O., and Alves, C. (2012). Software ecosystems: Trends and impacts on software engineering. In *Software Engineering (SBES), 2012 26th Brazilian Symposium on*, pages 206–210.

Santos, R. P. (2014). Reuseseem: An approach to support the definition, modeling, and analysis of software ecosystems. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 650–653, New York, NY, USA. ACM.

Santos, R. P. and Werner, C. (2011a). Treating business dimension in software ecosystems. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 197–201, New York, NY, USA. ACM.

Santos, R. P. and Werner, C. M. L. (2010). Revisiting the concept of components in software engineering from a software ecosystem perspective. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 135–142, New York, NY, USA. ACM.

Santos, R. P. and Werner, C. M. L. (2011b). A proposal for software ecosystem engineering. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 40–51. CEUR-WS.

Sasso, T. and Lanza, M. (2013). A closer look at bugs. In *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on*, pages 1–4.

Satyanarayanan, M. (2013). Cloudlets: At the leading edge of cloud-mobile convergence. In *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*, QoSA '13, pages 1–2, New York, NY, USA. ACM.

Sawyer, S. (2000). Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, 9(1):47–58.

Scacchi, W. (2007a). Free/open source software development: recent research results and emerging opportunities. In *The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers*, ESEC-FSE companion '07, pages 459–468, New York, NY, USA. ACM.

Scacchi, W. (2007b). Free/open source software development: Recent research results and methods. In Zelkowitz, M. V., editor, *Architectural Issues*, volume 69 of *Advances in Computers*, pages 243 – 295. Elsevier.

Scacchi, W. (2010a). Collaboration practices and affordances in free/open source software development. In Mistrík, I., van der Hoek, A., Grundy, J., and Whitehead, J., editors, *Collaborative Software Engineering*, pages 307–327. Springer Berlin Heidelberg. 10.1007/978-3-642-10294-3_15.

Scacchi, W. (2010b). The future of research in free/open source software development. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 315–320, New York, NY, USA. ACM.

Scacchi, W. and Alspaugh, T. A. (2012). Understanding the role of licenses and evolution in open architecture software ecosystems. *Journal of Systems and Software*, 85(7):1479 – 1494.

Schaeffer, D. J., Herricks, E. E., and Kerster, H. W. (1988). Ecosystem health: I. measuring ecosystem health. *Environmental Management*, 12(4):445–455.

Schmerl, B., Garlan, D., Dwivedi, V., Bigrigg, M. W., and Carley, K. M. (2011). Sorascs: a case study in soa-based platform design for socio-cultural analysis. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 643–652, New York, NY, USA. ACM.

Schmid, K. (2013). Variability support for variability-rich software ecosystems. In *Product Line Approaches in Software Engineering (PLEASE), 2013 4th International Workshop on*, pages 5–8.

Schneider, K., Meyer, S., Peters, M., Schliephacke, F., Mörschbach, J., and Aguirre, L. (2010). Feedback in context: Supporting the evolution of it-ecosystems. In Ali Babar, M., Vierimaa, M., and Oivo, M., editors, *Product-Focused Software Process Improvement*, volume 6156 of *Lecture Notes in Computer Science*, pages 191–205. Springer Berlin / Heidelberg. 10.1007/978-3-642-13792-1_16.

Scholten, U., Fischer, R., and Zirpins, C. (2012). The dynamic network notation: harnessing network effects in paas-ecosystems. In *Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners*, SIMPLEX '12, pages 25–30, New York, NY, USA. ACM.

Schugerl, P., Rilling, J., Witte, R., and Charland, P. (2009). A quality perspective of software evolvability using semantic analysis. In *Semantic Computing, 2009. ICSC '09. IEEE International Conference on*, pages 420 –427.

Schultis, K.-B., Elsner, C., and Lohmann, D. (2013). Moving towards industrial software ecosystems: Are our software architectures fit for the future? In *Product Line Approaches in Software Engineering (PLEASE), 2013 4th International Workshop on*, pages 9–12.

Schuur, H. v. d., Jansen, S., and Brinkkemper, S. (2011). The power of propagation: on the role of software operation knowledge within software ecosystems. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 76–84, New York, NY, USA. ACM.

Schwarz, N., Lungu, M., and Robbes, R. (2012). On how often code is cloned across repositories. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 1289–1292.

Schütz, S., Kude, T., and Popp, K. (2013). The impact of software-as-a-service on software ecosystems. In Herzwurm, G. and Margaria, T., editors, *Software Business. From Physical Products to Software Services and Solutions*, volume 150 of *Lecture Notes in Business Information Processing*, pages 130–140. Springer Berlin Heidelberg.

Seichter, D., Dhungana, D., Pleuss, A., and Hauptmann, B. (2010). Knowledge management in software ecosystems: software artefacts as first-class citizens. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, pages 119–126, New York, NY, USA. ACM.

Seidl, C. and Aßmann, U. (2013). Towards modeling and analyzing variability in evolving software ecosystems. In *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, VaMoS '13, pages 3:1–3:8, New York, NY, USA. ACM.

Seidl, C., Schaefer, I., and Aßmann, U. (2013). Capturing variability in space and time with hyper feature models. In *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, VaMoS '14, pages 6:1–6:8, New York, NY, USA. ACM.

Shaw, M. (2003). Writing good software engineering research papers. *Mini-tutorial for Proc ICSE" 03*.

Spauwen, R. and Jansen, S. (2013). Towards the roles and motives of open source software developers. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 69–80. CEUR-WS.org.

SSI (2014a). Fælles medicinkort - statens serum institut. http://www.ssi.dk/fmk. Accessed August 2014.

SSI (2014b). National Serviceplatform (NSP). http://www.ssi.dk/NSP. Accessed July 2014.

Statistics Denmark (2014). Statbank denmark. http://www.statbank.dk/statbank5a, Accessed July 2014.

Stefanuto, G., Spiess, M., Alves, A. M., and Castro, P. F. D. (2011). Quality in software digital ecosystems the users perceptions. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 85–88, New York, NY, USA. ACM.

Sundhed.dk (2014). Den offentlige sundhedsportal. https://www.sundhed.dk/. Accessed August 2014.

Sundhedsstyrelsen (2011). Aktivitet på private sygehuse 2006-2010. Sundhedsdokumentation.

Syed, S. and Jansen, S. (2013). On clusters in open source ecosystems. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 19–32. CEUR-WS.org.

Syeed, M. M., Hansen, K. M., Hammouda, I., and Manikas, K. (2014). Socio-technical congruence in the ruby ecosystem. In *Proceedings of the 10th International Symposium on Open Collaboration OpenSym*.

Taylor, R. N. (2013). The role of architectural styles in successful software ecosystems. In *Proceedings of the 17th International Software Product Line Conference*, SPLC '13, pages 2–4, New York, NY, USA. ACM.

Tchoua, R., Choi, J., Klasky, S., Liu, Q., Logan, J., Moreland, K., Mu, J., Parashar, M., Podhorszki, N., Pugmire, D., and Wolf, M. (2013). Adios visualization schema: A first step towards improving interdisciplinary collaboration in high performance computing. In *eScience (eScience), 2013 IEEE 9th International Conference on*, pages 27–34.

Teixeira, J. and Lin, T. (2014). Collaboration in the open-source arena: The webkit case. In *Proceedings of the 52Nd ACM Conference on Computers and People Research*, SIGSIM-CPR '14, pages 121–129, New York, NY, USA. ACM.

TELEKAT (2007). TELEKAT: Telehomecare, chronic patients and the integrated healthcare system. http://www.telekat.eu/.

Telesår (2006). Telemedicinsk sårvurdering. http://www.pleje.net.

The Economist (2011). Future-proofing western europe's healthcare – a study of five countries. The Economist Intelligence Unit.

The Economist (2014). The quantified self: Counting every moment. http://www.economist.com/node/21548493, Accessed July 2014.

Tymchuk, Y., Mocci, A., and Lanza, M. (2014). Collaboration in open-source projects: Myth or reality? In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 304–307, New York, NY, USA. ACM.

Valenca, G. (2013). Requirements negotiation model: A social oriented approach for software ecosystems evolution. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 393–396.

Vasilescu, B., Serebrenik, A., Goeminne, M., and Mens, T. (2014). On the variation and specialisation of workload—a case study of the gnome ecosystem community. *Empirical Software Engineering*, 19(4):955–1008.

Ververs, E., van Bommel, R., and Jansen, S. (2011). Influences on developer participation in the debian software ecosystem. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 89–93, New York, NY, USA. ACM.

ViewCare (2011). Viewcare. http://www.viewcare.com/.

Viljainen, M. and Kauppinen, M. (2011). Software ecosystems: A set of management practices for platform integrators in the telecom industry. In Regnell, B., Weerd, I., Troyer, O., Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M. J., and Szyperski, C., editors, *Software Business*, volume 80 of *Lecture Notes in Business Information Processing*, pages 32–43. Springer Berlin Heidelberg. 10.1007/978-3-642-21544-5_4.

Waltl, J., Henkel, J., and Baldwin, C. (2012). Ip modularity in software ecosystems: How sugarcrm's ip and business model shape its product architecture. In Cusumano, M., Iyer, B., and Venkatraman, N., editors, *Software Business*, volume 114 of *Lecture Notes in Business Information Processing*, pages 94–106. Springer Berlin Heidelberg.

WEA (2014). WEA - Workshop Ecosystem Architectures | 2nd International Workshop on Software Ecosystem Architectures (WEA 2014). http://wea.github.io/ Accessed August, 2014.

Weiss, M. (2011). Economics of collectives. In *Proceedings of the 15th International Software Product Line Conference, Volume 2*, SPLC '11, pages 39:1–39:8, New York, NY, USA. ACM.

Widjaja, T. and Buxmann, P. (2011). Compatibility of software platforms. In Heinzl, A., Buxmann, P., Wendt, O., and Weitzel, T., editors, *Theory-Guided Modeling and Empiricism in Information Systems Research*, pages 15–41. Physica-Verlag HD. 10.1007/978-3-7908-2781-1_2.

Wnuk, K., Manikas, K., Runeson, P., Lantz, M., Weijden, O., and Munir, H. (2014a). Evaluating the governance model of hardware-dependent software ecosystems – a case study of the axis ecosystem. In Lassenius, C. and Smolander, K., editors, *Software Business. Towards Continuous Value Delivery*, volume 182 of *Lecture Notes in Business Information Processing*, pages 212–226.

Wnuk, K., Runeson, P., Lantz, M., and Weijden, O. (2014b). Bridges and barriers to hardware-dependent software ecosystem participation – a case study. *Information and Software Technology*, 56(11):1493 – 1507.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer.

World Health Organization (2010). Telemedicine. opportunities and developments in member states. Global Observatory for eHealth series – Volume 2.

Wright, J. L., McQueen, M., and Wellman, L. (2013). Analyses of two end-user software vulnerability exposure metrics (extended version). *Information Security Technical Report*, 17(4):173 – 184. Special Issue: {ARES} 2012 7th International Conference on Availability, Reliability and Security.

Wynn, Donald, J. (2012). The evolving structure and function of commercial open source software ecosystems. In Cusumano, M., Iyer, B., and Venkatraman, N., editors, *Software Business*, volume 114 of *Lecture Notes in Business Information Processing*, pages 285–290. Springer Berlin Heidelberg.

Yin, R. K. (2013). *Case study research: Design and methods*. SAGE, fifth edition.

Yu, E. and Deng, S. (2011). Understanding software ecosystems: A strategic modeling approach. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 65–76. CEUR-WS.

Yu, L. (2011). Coevolution of information ecosystems: a study of the statistical relations among the growth rates of hardware, system software, and application software. *SIGSOFT Softw. Eng. Notes*, 36(6):1–5.

Yu, L. (2013). The market-driven software ecosystem. *IT Professional*, 15(5):46–50.

Yu, L., Ramaswamy, S., and Bush, J. (2007). Software evolvability: An ecosystem point of view. In *Software Evolvability, 2007 Third International IEEE Workshop on*, pages 75 –80.

Yu, L., Ramaswamy, S., and Bush, J. (2008). Symbiosis and software evolvability. *IT Professional*, 10(4):56 –62.

Yu, S. and Woodard, C. (2009). Innovation in the programmable web: Characterizing the mashup ecosystem. In Feuerlicht, G. and Lamersdorf, W., editors, *Service-Oriented Computing – ICSOC 2008 Workshops*, volume 5472 of *Lecture Notes in Computer Science*, pages 136–147. Springer Berlin / Heidelberg. 10.1007/978-3-642-01247-1_13.

Part II

Papers

Paper 1

Software ecosystems – A systematic literature review

Manikas, K. and Hansen, K. M. (2013c). Software ecosystems – a systematic literature review. *Journal of Systems and Software*, 86(5):1294 – 1306

The Journal of Systems and Software 86 (2013) 1294–1306

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Software ecosystems – A systematic literature review

Konstantinos Manikas*, Klaus Marius Hansen

Department of Computer Science (DIKU), University of Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 28 March 2012
Received in revised form 8 December 2012
Accepted 8 December 2012
Available online 20 December 2012

Keywords:
Software ecosystems
Software ecosystem
Systematic literature review

ABSTRACT

A software ecosystem is the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services. Arguably, software ecosystems are gaining importance with the advent of, e.g., the Google Android, Apache, and Salesforce.com ecosystems. However, there exists no systematic overview of the research done on software ecosystems from a software engineering perspective. We performed a systematic literature review of software ecosystem research, analyzing 90 papers on the subject taken from a gross collection of 420. Our main conclusions are that while research on software ecosystems is increasing (a) there is little consensus on what constitutes a software ecosystem, (b) few analytical models of software ecosystems exist, and (c) little research is done in the context of real-world ecosystems. This work provides an overview of the field, while identifying areas for future research.

1. Introduction

It has recently been suggested that *software ecosystems* (SECOs) are an effective way to construct large software systems on top of a software platform by composing components developed by actors both internal and external (Bosch, 2009; te Molder et al., 2011). In this setting, software engineering is spread outside the traditional borders of software companies to a group of companies, private persons, or other legal entities.

This differs from traditional outsourcing techniques in that the initiating actor does not necessarily own the software produced by contributing actors and does not hire the contributing actors. All actors, however, coexist in an interdependent way, an example being the iOS ecosystem in which Apple provides review of and a platform for selling applications in return for a yearly fee and 30% of revenues of application sale.[1] This is a parallel to natural ecosystems where the different members of the ecosystems (e.g., the plants, animals, or insects) are part of a food network where the existence of one species depends on the rest.

In addition to iOS, Google's Android ecosystem is a prominent example of a (smartphone) software ecosystem. Such ecosystems are arguably gaining importance commercially: it is, e.g., estimated that in 2012, more smartphones than personal computers will be sold.[2]

While software ecosystems are thus arguably gaining importance, research in software ecosystems is in its infancy, starting in 2005 with Messerschmitt and Szyperski (2005) and now with a dedicated workshop in its third year.[3] Our own literature search (see Section 3) revealed a gross list of 420 published papers on software ecosystems. However, until now there has been no systematic literature review (SLR) of the research literature on software ecosystems, leading to potential issues in identifying research gaps and contributions.

In the context of this, we have conducted a systematic literature review in the field of software ecosystems using the approach of Kitchenham and Charters (2007). As such, the purpose of this literature review is to provide an overview of the research reported in the field and identify possible issues that existing literature is not addressing adequately. This work is intended to function as a snapshot of the research in the field by (i) identifying and analyzing the different definitions of SECOs, (ii) analyzing the growth in research reported per year, (iii) classifying the research by type of result, (iv) defining and analyzing the software architecture and structure of SECOs, and (v) analyzing to which extent research is connected to SECO industry.

1.1. Article structure

The rest of this article is organized as following: in Section 2 we specify the review protocol, in Section 3 we document the extraction of the literature, in Section 4 we analyze the literature and answer the research questions, in Section 5 we list possible threats

* Corresponding author. Tel: +45 23839917.
E-mail addresses: kmanikas@diku.dk (K. Manikas), klausmh@diku.dk (K.M. Hansen).

[1] http://developer.apple.com/programs/ios/distribute.html.
[2] http://www.slideshare.net/CMSummit/ms-internet-trends060710final.
[3] http://www.softwareecosystems.org/workshop/.

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.12.026

to the validity of this work and identify areas not covered from the literature and in Section 6 we conclude.

2. Review protocol

The applied review protocol is based on the guidelines of Kitchenham and Charters (2007). The establishment of the review protocol is necessary to ensure that the literature review is systematic and to minimize researcher bias. As such, the literature review is focused on a set of research questions that serve the aim of this work and derive from the reasons that initiated this review. The review protocol is organized in a way that the research questions define the main areas this study is focusing on. Section 2.2 defines the paper literature extraction strategy including the list of resource libraries, the search query and inclusion/exclusion criteria.

2.1. Research questions

The purpose of this systematic literature review is to provide an overview of the research reported in the field of SECO. In this overview, we intent to address the following research questions:

RQ 1: *How is the term 'software ecosystem' defined?*
In order to be able to analyze the field of SECOs, we should first define the SECO as object of study. Thus, the first objective of this work is to provide an overview of how the research community defines the term 'software ecosystem'. We achieve that by looking into the SECO definitions in the literature and comparing them. This will create an understanding of what the research community means by the term SECO.

RQ 2: *What is the research output per year in the SECO field?*
By grouping the literature per publication year we are able to identify possible trends in the research invested in the field of SECOs. An increase in the number of publications per year, for example, would imply the increase in importance of the field while a decrease in the number of publications might have as a possible reason the research in the field reaching a dead end. Analyzing the trends might give an idea of how the importance of the field of SECOs is changing with time.

RQ 3: *What is the type of result that software ecosystem research reports?*
After having defined the term SECO, a question that we want to address is what kind of research this field reports. Therefore, it is of interest to classify the papers according to the contribution they make. From a software engineering perspective, Shaw's classification of research results (Shaw, 2003) has been chosen. The classification contains the following categories:

Procedure or technique: This category includes papers that are providing a concrete and implementable way to solve a SECO problem. The solutions should be in the form of a procedure or technique that can be applied and not general rules of thumb or reported experiences. For example, Kazman et al. (2012) analyze a series of traditional software design and software architecture principles and methods in the perspective of the SECOs (or software-intensive ecosystems as they are called in the paper). This results in some new or adapted methods for the software design and architecture of these software-intensive ecosystems.
Qualitative or descriptive model: Papers using models based on qualitative analysis of data or well argumentation of existing cases. Papers in this category provide an analytical or descriptive model for the problem area. As an example the analysis of two different kinds of SECO: the "as-a-service" and "on-premise" software ecosystems that derived from a comparative study of two existing SECOs presented in Hilkert et al. (2010).
Empirical model: This category includes papers that use models derived from the quantitative data collection of the problem area. A paper of this category studies empirical data and concludes some analysis or predicting model. For example, Yu et al. (2008) extract information from open source systems to assess the evolvability of software.
Analytic model: Papers using models based on automatic or mathematical manipulation for solving a specific problem. For example the paper of Capuruço and Capretz (2010) that propose a prediction of recommendations and interaction between the members of a social ecosystem based on a mathematical analysis of the member relationships.
Tool or notation: A tool or notation created or implemented applying some method or technique. For example, a tool for recovering components and their relationships in free or open source projects, proposed by Lungu (2008)
Specific solution, prototype, answer, or judgment: Papers documenting a complete solution, evaluation of a theory or comparison of different theories based on a software engineering problem. The result is addressing a specific problem. An example would be Pettersson and Gil (2010) who address reusability and adaptability issues in mobile learning systems
Report: Papers documenting knowledge and experience obtained, rules of thumb or checklists but not systematic enough to be a descriptive model. For example, the analysis of the hybrid business and revenue models that software companies can have (Popp, 2011).

RQ 4: *What is the role of architecture in software ecosystem research?*
For single systems, software architecture is seen as important in determining the quality of a system being built (Bass et al., 2003; Hansen et al., 2011). In relation to this, we analyze the extent to which SECO literature stresses software architecture. We evaluate the literature in whether it is documenting any considerations towards SECO software architecture. In doing so, our concept of software architecture is in line with Bass et al. (2003):

> "The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them."

We here extend the definition to concern software ecosystems, i.e., we define 'software ecosystem architecture' as the structure or structures of the software ecosystem in terms of elements, the properties of these elements, and the relationships among these elements. The SECO elements can be systems, system components, and actors. Relationships then include software architecture-related relationships as well as actor-related relationships such the relationship between two actors.

RQ 5: *How is the connection between research and industry in the area of software ecosystems?*
It is of interest to know how close industry and research are in the field of software ecosystems. Research benefits from realism of problems when connected to the industry while industry arguably may become more innovative and efficient when connected to research. In the case of SECOs research results are more valid when they are concerning existing SECOs, while studies of problems in existing SECO can help the industry improve.

We investigate how connected the research world is with the industry by examining how much of the literature has focused on real-world SECOs. We accept that a paper has focus on a real-world SECO when it either presents an existing SECO as an object of study or uses the data from the study of one to support a claim or result. For example, this could be a paper that is deducting information of the external actors of an ecosystem by studying the relationships between the actors of one or more existing SECOs. However, we do not include papers that merely mention a SECO, e.g., in order to support their definition of SECOs, and that thus present no study of the SECO.

2.2. Defining the literature body

The strategy for collecting the relevant literature is twofold: (i) a keyword search in a list of scientific libraries and (ii) the collection of the papers from the SECO workshop series.

With respect to (i), the scientific libraries included in the search are:

1. The ACM Digital Library[4]
2. IEEE Explore[5]
3. Springer Verlags' digital library, SpingerLink[6]
4. ScienceDirect.[7] An online collection of published scientific research operated by the publisher Elsevier.
5. Thomson Reuters' Web of Science.[8] An online academic citation index.

The literature extraction consists of two separate keyword searches with the search terms "software ecosystem" and "software ecosystems" in the libraries above. The search query is intentionally kept simple so we can extract the maximum number of papers containing the terms. We specifically define SECO(s) as the keyword to underline the differentiation of the field of software ecosystem from different ecosystems like business, digital or social. The borders of the SECO field can be sometimes vaguely defined especially when overlapping with other kinds of ecosystems. For example some SECOs in the literature can also fit in a digital ecosystem definition and there are several studies on business ecosystems that produce software. The purpose of this work, however, is to study software ecosystems and any possible intersections with other ecosystems should be studied from the SECO point of view. Therefore, this study does not include studies on other kinds of ecosystems.

With respect to (ii), we include the papers from the International Workshops on Software Ecosystems (IWSECO).

The selected literature body collected from both (i) and (ii) should commit to a set of inclusion criteria:

- The literature should address software ecosystems as an area of research, either main or secondary. Therefore, the keywords "software ecosystem(s)" should exist as a whole and continuously in at least one of the fields: title, keywords or abstract. Additionally, possible composites of the keywords should be examined, e.g., software-intensive ecosystems.
- Be research papers, i.e., being published in a scientific peer reviewed venue.
- Be written in English.
- Have a document body that is more than one page long.

Consequently, the literature does not contain books, extended abstracts, presentations, presentation notes, keynotes or papers written in other language than english.

The literature body is the results of the following steps:

1. Collecting all the literature. The literature collection is the combination of the scientific library search and the IWSECO papers. The library search, at this point includes a search of the keywords in the whole text body in order to include the maximum amount of papers.
2. Applying inclusion/exclusion criteria. The literature collection resulting from the previous step are searched for the keywords in the fields title, abstract, keywords.
3. Verifying rejected papers. The rejected literature from the previous step is searched for only the terms "ecosystem(s)" and "software" in the fields title, abstract, keywords and evaluated if they are related literature. This would avoid rejecting papers with different combinations of the keywords, for example "software-intensive ecosystems".
4. Verifying included papers. The included literature that resulted from the two previous steps is verified manually by reading the abstract and conclusion. In this step, we make sure that the papers included in the review provide results that are directly or indirectly related to the field of SECO.

3. Collecting the literature body

To obtain the literature body of our review, we apply the systematic literature review (SLR) protocol described in Section 2 with the extraction date of June 11, 2012. The four steps for defining the literature body described in Section 2.2 can be seen in Table 1. The literature collection starts with 420 papers extracted from the libraries. All the IWSECO papers are included in this collection. After applying the inclusion/exclusion criteria, we reject 297 paper. Out of the 297 rejected, we apply step 3 and included six papers with key words "open ecosystems", "software-intensive ecosystems", "ERP ecosystems", "information ecosystem", "source code ecosystems", "Eclipse ecosystem". In step 4 we went through 129 papers (123 from step 2 plus 6 from step 3) and find 90 papers relevant. We contribute the high number of rejected papers in step 2 to two reasons: (i) some libraries would search in the whole paper text body and thus retrieve papers mentioning SECO but not reporting research on that field and (ii) Science Direct does not recognize the quotation marks in "software ecosystem" or "software ecosystems" so it retrieves results that the words are not adjacent to each other but in different locations in the texts, therefore there were many papers not related to software engineering. We also note that from the six papers selected in step 3, only one (Kazman et al., 2012) is part of the included papers.

During the data extraction process, we read the papers found relevant and extracted interesting information and information needed to address the research questions. The information extraction is in the form of descriptive text enclosed by identifying labels for automated sorting. In continuation, a set of custom scripts export the requested information.

Table 1
The steps and included papers to define the literature body.

Step	Nr of papers
1. Collecting the literature	420
2. Applying inclusion/exclusion criteria	123
3. Verifying rejected papers (included)	6
4. Verifying included papers	90

[4] http://dl.acm.org/.
[5] http://ieeexplore.ieee.org.
[6] http://www.springerlink.com/.
[7] http://www.sciencedirect.com/.
[8] http://apps.webofknowledge.com/.

4. Analysis

In this section we analyze the literature and the results of the review. The section is organized according to the research questions in Section 2.1.

4.1. Defining SECO

During this literature review, we obtained an overview of the general field referred to as *software ecosystems*. One of our initial aims was to define the term SECO by summarizing the definitions in the literature. Looking into the literature, our first remark is that we found a large number of papers (40 out of the total of 90) that did not define the term SECO. This is, either because the authors are basing their work on previous research (own or not) that would provide the background and definition or because the main focus of the paper is not in the general field of SECO. For example, Bosch (2010a) is not providing any definition, but he is referring back to his own work (Bosch, 2009) where he provides a definition and more detailed analysis of the field. On the other hand, Popp (2011) defines the business and revenue models for SECOs. In his paper, he is providing definitions for the business and revenue models that is the main focus, instead of a definition of a SECO. This, however, does not make it of less value to the research field of SECOs.

Taking the papers that provide a definition, we notice that few of them are defining the SECO with their own words. Two of these papers are also citing more definitions from the literature along with their own. The rest of the papers, are defining the field by using one or more definitions from the existing literature. When we analyzed the definitions, we found that we can group the quoted definitions in four groups according to the source of the definition:

Messerschmitt and Szyperski (2005) is the oldest definition of SECO in the found literature referring to the book on SECO published in 2005.

> "Traditionally, a software ecosystem refers to a collection of software products that have some given degree of symbiotic relationships." (Messerschmitt and Szyperski, 2005)

Jansen et al. (2009b) mainly refer to the following definition:

> "We define a software ecosystem as a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently under-pinned by a common technological platform or market and operate through the exchange of information, resources and artifacts." (Jansen et al., 2009b)

Bosch (2009) and Bosch and Bosch-Sijtsema (2010b,c) provide two definitions in their papers. The papers quoting his definitions are taking one of the following:

> " A software ecosystem consists of the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions." (Bosch, 2009)
>
> "A software ecosystem consists of a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs." (Bosch and Bosch-Sijtsema, 2010b,c)

Lungu et al. (2010a) are presenting a different definition of the SECOs that is adopted by a number of papers:

> "A software ecosystem is a collection of software projects which are developed and evolve together in the same environment." (Lungu et al., 2010a)

In Table 2 we show the different groupings and the papers belonging to each group. The in the column Papers refer to the literature body listed in Appendix A.

Not surprisingly, if we look at the definitions we can see that they have two things in common: they concern software in some form (software systems, products, services, or a software platform) and they are all including some kind of relationships either "symbiotic", "common evolution", "business" or "technical". If we look at what perspective the authors have in the definitions, we note that Messerschmitt and Lungu et al. have a pure technical perspective by talking about software and its symbiosis/co-existence, while Bosch et al. and Jansen et al. include, apart from the technical, a social and business perspective to their definition and the symbiosis is not only on the technical level. Taking the two wider-perspective definitions of Bosch et al. and Jansen, which are referenced by the majority of the papers that provide a definition for SECO (65%), we can identify three main elements in their definitions:

Common Software The software appears either as a "common technological platform" (Jansen et al., 2009b), "software solutions" (Bosch, 2009) or "software platform" (Bosch and Bosch-Sijtsema, 2010b,c)

Business This is expressed as either "a set of business" (Jansen et al., 2009b), "business ecosystem" (Bosch, 2009), a "community of users that have needs to be satisfied" (Bosch and Bosch-Sijtsema, 2010b,c). In this element, the term "Business" is implying a wider sense than the profit or revenue models. This element also includes possible benefits other than financial revenues, e.g., the benefits an actor would get from the involvement in an free or open source project.

Connecting Relationships "a set of businesses (. . .) together with the relationships among them " (Jansen et al., 2009b), "actors in the associated social ecosystem" (Bosch, 2009), "community of domain experts" and "community of users" (Bosch and Bosch-Sijtsema, 2010b,c)

Combining the definitions above with the three elements identified, we define a software ecosystem as the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services. Each actor is motivated

Table 2
The papers belonging to each group of SECO definition.

Definition	Papers	Total
Not available	[19, 1, 45, 39, 43, 2, 5, 6, 9, 11, 48, 49, 59, 52, 51, 54, 42, 53, 36, 46, 31, 22, 21, 35, 33, 26, 55, 32, 24, 63, 82, 74, 75, 69, 62, 64, 70, 78, 67, 83]	40
Jansen et al.	[3, 4, 10, 16, 13, 28, 37, 44, 14, 29, 12, 6, 27, 87, 86, 72, 76, 71, 61, 65, 66, 60, 90, 84]	24
Bosch et al.	[40, 41, 10, 13, 20, 23, 44, 14, 17, 12, 89, 77, 79]	13
Own	[38, 8, 30, 58, 56, 47, 12, 34, 73]	9
Lungu et al.	[7, 15, 18, 80, 68, 81]	6
Messerschmitt et al.	[40, 50, 37, 57, 85]	5

Table 3
Papers published per year.

Year	Papers	Total
2007	[31, 46, 57]	3
2008	[50, 53, 54]	3
2009	[9, 10, 11, 20, 6, 42, 51, 52, 56, 58]	10
2010	[1, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23,24, 27, 28, 29, 30, 33,34, 35, 36, 37, 38, 39,41, 43, 45, 47, 48, 49,55, 59]	32
2011	[2, 3, 4, 5, 6, 7, 8, 26,32, 40, 44, 87, 89, 88,73, 72, 74, 68, 71, 69,64, 61, 65, 70, 79, 66,60, 67, 83, 85, 90, 84]	32
2012	[63, 80, 86, 82, 76, 77,75, 62, 78, 81]	10

by a set of interests or business models and connected to the rest of the actors and the ecosystem as a whole with symbiotic relationships, while, the technological platform is structured in a way that allows the involvement and contribution of the different actors. In other words, the SECO provides possibilities for the actors to benefit from their participation in the ecosystem. The types of benefits might vary depending on the actor and the nature of the ecosystem. In a commercial ecosystem the actors might gain direct revenues, e.g., developers making apps for iPhone and selling them in the App Store, while in a non-commercial ecosystem the actors might participate for non-monetary benefits (fame, knowledge, ideology and so on), e.g., the developers contributing to Apache. Additionally, the actors' relationships to the ecosystem as a whole are of mutual interest (mutualism): the actors' benefits increase by the thriving of the ecosystem and the ecosystem benefits by increased actor activity. The relationships among the actos in a SECO, on the other hand, are characterized by the wider spectrum of symbiotic relationships. Depending on the actors and their activity, two actors might have mutual benefits (mutualism), be in direct competition (competition/antagonism), be unaffected (neutralism) or one being unaffected while the other is benefiting (amensalism) or harmed (parasitism) by their relationship.

When looking at the rest of the papers, we note that there is a number of papers that assist in the conceptualization of the field in a wider sense than just providing the definition of SECOs. These papers are used as a conceptual base of succeeding work. In this concept, Bosch (2009) proposes a taxonomy where he divides SECOs in three categories: operating system-centric, application-centric and end-user programming software ecosystems. In continuation he discusses the steps needed for the transition to a SECO and implications this transition might have. Jansen et al. (2009a), apart from providing the definition for SECO seen above, propose three scopes to study SECOs that are also explained briefly in Boucharas et al. (2009): an external view on ecosystems that studies the SECOs themselves and the markets around them, an internal view of a SECO that is focusing on software supply networks and their relationships, and an organization-centric perspective that studies the actors and their relationships. Campbell and Ahmed (2010) propose a view of SECO consisted of three dimensions: business, architectural and social. Dhungana et al. (2010) make a comparison of the SECO with biological ecosystems from the perspective of resource management and biodiversity and underline the importance of diversity, monitoring of health and supporting social interaction for the field of SECO.dos Santos and Werner (2011b) collect the concepts appearing in the papers from IWSECO 2009 to 2010 and organize them in three views: SECO architecture, SECO strategies and tactics and SECO social networks. Finally, Barbosa and Alves (2011) conduct a systematic mapping in the field of SECO and categorize the research in eight fields unfolded around open source software, ecosystem modeling, and business issues.

4.2. Yearly activity

Another point of study in this work, is the analysis of the year of publication. We order the papers according to their publication year as can be seen in Table 3. The literature on SECO starts in 2007 (although (Messerschmitt and Szyperski, 2005) dates back to 2005, it was excluded from this study for being a book and not a research paper). The first two years – 2007 and 2008 – provide an equally low number of papers. However, an increase appears in 2009 and continues to 2010 with 2010 and 2011 having the same amount of papers.

The increase of papers gives us a clear sign that the field of SECO is gaining in importance among the published research. This is also underlined with the establishment of a workshop dedicated to SECOs, the International Workshop on Software Ecosystems (IWSECO), in 2009. While this does not give insight into software ecosystems in themselves, it stresses the potential significance of the concept.

4.3. Research results

As noted in research question in Section 2.1, it is of interest to examine what kind of results the papers are reporting. We have classified the papers in the categories listed in research question in Section 2.1 and can be seen in Table 4. As it can be seen from the table, the majority of the papers fall under the Report category. This means that these papers have as contribution knowledge and experience obtained, rules of thumb or checklists or interesting observations but they are not systematic enough, nor generic enough to be applied to different domains or too abstract to provide a concrete contribution. An example of a paper falling under this category is the paper by Dhungana et al. (2010) that compares

Table 4
The papers grouped according to the result groups.

Result	Papers	% of total
Report	[19, 43, 2, 6, 10, 8, 59, 52, 56, 51, 36, 46, 31,20, 35, 12, 26, 32, 24,14, 29, 33, 17, 87, 73,86, 82, 76, 77, 71, 75,64, 61, 65, 70, 78, 66,67, 83, 85, 84]	44
Tool or notation	[9, 48, 58, 54, 15, 22,47, 6, 80, 68, 69, 62, 79,81]	15
Procedure or technique	[40, 41, 5, 11, 30, 53, 16, 13, 28, 18, 14,17]	13
Qualitative or descriptive model	[38, 39, 3, 4, 7, 21, 37, 29, 34, 74]	11
Empirical model	[45, 50, 44, 57, 55, 27, 89]	8
Analytic model	[1, 42, 63, 72, 90]	5
Specific solution	[49, 23, 33, 88]	4

Table 5
The papers according to the SECO Architecture groups.

SECO architecture group	Papers	% of total
SECO SE	[40, 19, 45, 38, 39, 43, 11, 49, 58, 36, 46, 15, 22, 47, 35, 29, 17, 57, 87, 63, 68, 69, 70, 79, 66, 81]	35
SECO business and management	[2, 4, 5, 10, 58, 56, 16, 20, 23, 37, 44, 14, 33, 12, 6, 27, 87, 88, 86, 76, 77, 71, 75, 62, 66, 60, 67, 83, 90]	39
SECO relationships	[3, 4, 6, 30, 13, 31, 21, 33, 17, 87, 73, 86, 82, 72, 76, 61, 65, 85, 84]	26

SECOs to the natural ecosystem and reports observations and a research agenda. This paper does not report any concrete method of some kind and the data used is not systematic enough for the paper to be included in the qualitative model.

Looking at the distribution, we note that the category with the most papers after Report is Tool or Notation. The papers of this category are implementing tools or notations that are mostly using data from FOSS SECOs. This, as we will discuss more in Section 4.5, is related to the fact that FOSS SECOs provide access to a lot of technical data, e.g., commit history or bug reports that are not easy to access in proprietary SECOs. The third category, Procedure or Technique, includes papers that report an implementable technique to solve a specific task. For example the paper by Fricker (2009), that proposes a technique for requirement management in SECOs.

When examining the percentage of papers that fall under each category, we can make the following observations. The field of SECOs is a new research field, with the first papers appearing in 2007. This implies that there is an amount of research resources spent in defining the field and its limits, for example the papers analyzed in Section 4.1. In addition, as it is shown in Section 4.5, there is a relatively small amount of research spent in examining SECOs in the industry. These two reasons result in the Report category having a bigger percentage to all the other categories. Additionally, we recognize that the field of SECO is wide and can have multiple research perspectives, such as software engineering (SE), social networks or technical management. In connections to this, there have been several papers focusing on some specific aspect of the field providing specific and implementable techniques. This potentially explains the high percentage in the Tool or Notation and Procedure or Technique categories.

4.4. SECO architecture

To address RQ in Section 2.1, we separated and analyzed the papers that are addressing the SECO architecture as defined in the research question. During the analysis of the papers, we could identify three logical groups of SECO architecture papers. Table 5 shows the distribution of the papers according to their main research focus. Below we elaborate on the three SECO architectural groups describing them in more detail. Papers used in the description of a group might not represent their main research focus.

4.4.1. SECO software engineering

Software ecosystems, having as a product one or several software systems have problems that belongs to the software engineering field. A part of the SECO literature is focusing on SE either by using SE practices directly or by adapting existing SE practices to the SECO context. This category consists of papers focusing on more technical issues related directly or indirectly to the technological platform of a SECO. It contains 26 papers, i.e., 35% of the literature focusing on SECO architecture aspects.

One important aspect of this category is software architecture. The software architecture of a SECO should support the nature of the ecosystem (i.e., be adapted to the needs of the specific SECO), follow the SECO management, business rules and restrictions and allow the integration and existence of multiple functionality in a secure and reliable manner. A modular and flexible architecture would allow integration and interoperability of the developed software (Viljainen and Kauppinen, 2011; Bosch, 2009). Interfaces allow external development on a SECO platform. The stability and translucency of the platform interfaces are essential for the component integration and interaction (Cataldo and Herbsleb, 2010; Bosch, 2010a). Changes to existing interfaces or components might create inconsistencies to dependent components (Robbes and Lungu, 2011; Lungu et al., 2010a,b). Process-centric approaches are not effective in managing large scale software, instead system architecture should be used as a coordination mechanism (Bosch and Bosch-Sijtsema, 2010a). Constantly evolving software requires the adaptation of the software development processes. Development should be integration-centric, independent deployment and releases should be organized in a release grouping and release train fashion (Bosch and Bosch-Sijtsema, 2010b; Bosch, 2010a). Architectural design and analysis techniques are based on a set of principles as identifying business goals, describing architectural significan requirement, tactics and architectural evaluation. These principles are used in defining the software architecture of a SECO (Kazman et al., 2012).

Apart from software architecture, in the wider SE related subjects, requirement elicitation appears as an interesting challenge in the SECO concept as the stakeholders are multiple and distant from the central ecosystem management. The use of "requirement value chain" is proposed to propagate requirements (Fricker, 2009, 2010).

4.4.2. SECO business and management

This category contains papers focusing on the business, organizational and management aspects of SECOs. Independently of how each SECO is organized, there is an organizational and management entity that is responsible for monitoring, operational and decision making part of the SECO whether it being a proprietary company, an open source community or a hybrid of the two. This category is sub-divided into two groups: organizational & management and business.

The organizational and management group includes papers that are focusing on the organizational actions in a SECO. These actions are initiated from decisions, rules and processes or controlling mechanisms. The main activities of this group are summarized in: monitoring the SECO, evaluating and decision making, and taking actions.

In order to ensure that a SECO is functioning well, specific measurements need to be introduced that would provide an overview of the state of the SECO while at the same time raise attention for actions and allow comparison of SECOs. The literature is referring to the concept of the *health of a software ecosystem* (van Ingen et al., 2011; van Angeren et al., 2011; van den Berk et al., 2010; dos Santos and Werner, 2011a,b; Kilamo et al., 2012; Jansen et al., 2012, 2009a; Viljainen and Kauppinen, 2011; Mizushima and Ikawa, 2011; McGregor, 2010; Dhungana et al., 2010; Boucharas et al., 2009). This concept has been introduced by Iansiti et al. as a way to measure the performance of a business ecosystem (BECO). In more detail they measure the "*extent to which an ecosystem as a whole is durably growing opportunities for its members and those*

who depend on it" (Iansiti and Levien, 2004a) and inspired from biological ecosystems define the health of a (business) ecosystem as an analogy to robustness, productivity and niche creation (Iansiti and Levien, 2004b,a). These studies, although excluded from the collected literature, are referenced by the majority of the literature elaborating on SECO health (van Ingen et al., 2011; van Angeren et al., 2011; van den Berk et al., 2010; Kilamo et al., 2012; Jansen et al., 2012, 2009a; Viljainen and Kauppinen, 2011; Mizushima and Ikawa, 2011; McGregor, 2010; dos Santos and Werner, 2011b; Boucharas et al., 2009). An additional study on the health of business ecosystems that is referenced by several papers of the literature (van Angeren et al., 2011; van den Berk et al., 2010; Kilamo et al., 2012; Jansen et al., 2012, 2009a) that are elaborating on SECO health, is the paper of den Hartigh et al. (2006) that, based on the Iasiti et al studies mentioned above, applies health measurement to Dutch IT business ecosystems. In the SECO field, van den Berk et al. (2010) base their work on BECO health to create a strategy assessment model.

The proper evaluation of SECO measurements, such as health, supports and encourages correcting or improving actions in the SECO. This requires a management entity that would have the power and possibility to apply changes both in the technical but also in the organizational aspects of the SECO. To our knowledge there is no study in the SECO literature on the different management entities and the decision making mechanisms applied to drive the SECO. This might be because of high variability in management models or the disclosure of information in proprietary SECOs. It would be possible to study the decision making mechanisms of a FOSS project where changes are applied, e.g., based on online member voting, but it is challenging to study how a proprietary SECO canalizes information from the peripheral actors, evaluates this information and decides on actions based on that.

After monitoring the SECO and concluding in a set of decisions, a next step is to execute these decisions. One of the ways of applying actions that appears in the literature is communication. A clear view on the direction that the ecosystem would evolve and the communication of this view to the ecosystem actors and involved parties is underlined as a necessity (Bosch, 2009; Viljainen and Kauppinen, 2011). Creating *roadmaps*, *visions* or long-term *strategic planning* of the ecosystem allows the actors to plan, in their turn, their activity in the ecosystem and align their business models with the SECO roadmaps (Kakola, 2010; Bosch, 2009; Hanssen, 2011; Viljainen and Kauppinen, 2011; Jansen et al., 2012; van den Berk et al., 2010). At the same time the ecosystem can set the requirement of the SECO actors to commit to the published roadmaps (Bosch and Bosch-Sijtsema, 2010b,c). From a more practical perspective, the ecosystem orchestrators can organize the component composition by providing a long-term plan of organized releases in a *release management* or *release trains* that the actors can coordinate with (Bosch and Bosch-Sijtsema, 2010b,c; Fricker, 2010; Jansen et al., 2012; van den Berk et al., 2010; van der Schuur et al., 2011; Bosch, 2009). Bosch and Bosch-Sijtsema (2010c) analyzed the concept of *release grouping* where different groups of components are released in different times allowing less coordination and communication overhead. Kilamo et al. (2012) introduce the *release readiness assessment* where proprietary software is assessed on its ability to be released as open source/ open ecosystem.

An important part of the SECO business and management category is related to the business perspective of the ecosystem. As explained in the definition analysis, the business perspective is important as without a solid business and business model serving the SECO and its actors, the SECO might lose its actors to competitive businesses or ecosystems and risk extinction. It is essential to underline that the business and business model as mentioned here do not necessarily imply monetary benefits. The business model that would serve the SECO actors, as mentioned in the definition, might imply value in other forms, for example fame or experience in the case of a FOSS SECO actor. The same applies to the SECO itself. A SECO might include other benefits than revenues in its business model. An example would be advantage over competitors or "visibility within the market" (van Angeren et al., 2011). This implies that the traditional software company business models where the revenues are a result of software license selling cannot be fully applied in the ecosystem concept. Popp (2011) provides an analysis of business models that are applied in three ecosystems and makes a separation between the business models and the revenue models of a SECO. He underlines the importance of revenue models and states that "*revenue models (. . .) often containing one or more non-monetary compensations, can be a source of competitive advantage*" (Popp, 2011). Burkard et al. (2012) refer to revenue models from two perspectives: actors or niche players provide their products for a fee and the SECO orchestrator or hub requires a fee from the actors. This fee can be base either on fixed or variable price models.

Although, selling software licenses might not be a main revenue venue for a SECO, the issue of software licenses is still of interest in the SECOs. SECOs collect code developed by different developers or companies with different policies and many times even in an combination of proprietary and open source. Addressing or avoiding possible intellectual property right (IPR) or licensing violations would ease the software integration, allow possible reuse that might lead to more niche creation, clarify possible business models and avoid legal complications that demand heavy resources. Licensing and IPR issues appear in a number of papers (Alspaugh et al., 2009; Jansen et al., 2012, 2009b; Mizushima and Ikawa, 2011; te Molder et al., 2011; Kilamo et al., 2012; Scacchi and Alspaugh, 2012) in the literature. In relation to this, Alspaugh et al. (2009) and Scacchi and Alspaugh (2012) discuss the issue of software licensing in open architecture systems, recognize changes in licenses on different versions of the same component or in the evolution of a software system and propose a structure for modeling software licenses. Mizushima and Ikawa (2011) analyze the IP management process of Eclipse called the "Eclipse Legal Process" and state that this process was a reason for vendors to join Eclipse. Anvaari and Jansen (2010) analyze the mobile software platforms and evaluate their level of openness taking into consideration also their licensing policies. Finally, Popp (2011) names three roles in the intellectual property (IP) business utilization: the IP distributors that sell IPR from the inventors or usage rights to the customers, the IP lessors that "rents" IPs or products of IP (e.g., software) for a specific time and the IP brokers that matches the needs of an IP requestor to an IP owner. For example an IP broker might facilitate a startup software company to find software vendors.

4.4.3. SECO relationships

An open technological platform in combination with a set of management processes and business models, cannot create a SECO without the social aspect. A community, social network or a set of actors weaved around a platform and sets of rules communicating and interacting both among themselves and with the platform is essential. Because of the existence of this interaction, the software architecture of the platform has to be designed with different considerations than a proprietary platform. The management process, business models and IPR issues become more complicated while at the same time the evolution of the system is faster and towards several directions while the SECO gains privileged position in the market. There are several actors that might be part of a SECO. The following list gives an overview of the most common actors encountered in the literature.

Orchestrator [9], "keystone (player, organization)", [10] "hub",[11] "shaper",[12] "management (unit)",[13] or "platform owner"[14] is a company, department of a company, actor or set of actors, community or independent entity that is responsible for the well-functioning of the SECO. This unit is typically managing the SECO by running the platform, creating and applying rules, processes, business procedures, setting and monitoring quality standards and/or orchestrating the SECO actor relationships.

Niche player [15]"influencer",[16] or "component developer/builder/team",[17] is the SECO actor that contributes to the SECO by typically developing or adding components to the platform, producing functionality that customers require. This actor is part of the SECO and complements the work of the keystone by providing value to the ecosystem. Depending on the management model of the ecosystem the niche players might influence the decision making in the management of the SECO.

External actor [18]"external developer (team)",[19] "third party developers/community",[20] "external parties",[21] "external partner",[22] "external entities",[23] "participant",[24] or "external adopter",[25] is the actor (company, person, entity) that makes use of the possibilities the ecosystem provides and thus providing indirect value to the ecosystem. This actor is external to the SECO management and usually has an activity limited to the actor's interest. Depending on the nature of the ecosystem, the external actor might be developing on top of or parallel to the SECO platform, identify bugs, promote the SECO and its products or propose improvements. This type of actor includes the role of the participant or follower in FOSS SECOs. An actor that is member of the SECO with either participation of limited responsibility or simply observing the evolution of the SECO from the inside.

Vendor "independent software vendor (ISV)",[26], "reseller" or "value-added reseller (VAR)",[27] is mainly the company or business unit that makes profit from selling the products of the SECO to customers, end-users or other vendors/VARs. The products might be complete integrations, components, selling or leasing of licenses or support agreements. A vendor that is modifying the SECO product by, e.g., adding functionality or combining different components together is called VAR.

Customer or "end user" is the person, company, entity that either purchases or obtains a complete or partial product of the SECO or a niche player either directly from the SECO/niche player or through a vendor/VAR.

A different characterization of the social network of a SECO appears in (Jansen et al., 2012; Scacchi and Alspaugh, 2012) where they characterize the SECO niche as a software supply network of producers, integrators and customers.

An interesting perspective of SECO relationships is the actor participation model that SECOs follow. Different ecosystems apply different models for allowing actors to contribute to the ecosystem. These models are many times related to the nature of the platform and to what extent it allows/supports different kinds of collaboration, but mostly to the business model behind the ecosystem. To explain this better, we take the actor participation model of three ecosystems as an example: a traditional FOSS project that is often open to any participant willing to join, the Eclipse ecosystem where developers can join freely but have to go through the Eclipse Legal Process every time they commit code (Mizushima and Ikawa, 2011) and the the Open Design Alliance (ODA) where actors have to pay an annual fee to be part of the ecosystem (van Angeren et al., 2011). The *openness* or *closeness* of a SECO describes how easy it is for an actor to participate in an ecosystem. The measurement of the openness of a SECO is an interesting perspective that affects the social network of an ecosystem. As already mentioned, the level of openness depends on parameters outside of the SECO social network perspective, however, it is analyzed as part of this perspective since it affects heavily the social networks. te Molder et al. (2011) claim that the openness and closeness of a platform is not binary, but there are many different levels. In their paper they introduce the concept of "clopeness" and propose a model for assessing the clopeness of a SECO. Jansen et al. (2012) state that the complicity of opening or closing the SECO as "multi-facet and cannot be judged without extensive study". They also explain that the benefits of opening up the ecosystem are often not clear, while a post-evaluation of whether the ecosystem was ready for the changes will be reflected in the SECO health after the changes have been applied. Finally they make a separation between the supply and demand of a SECO and mention that a SECO can choose to open either of them or both.

In the software supply network, Riis and Schubert (2012) analyze how the relationships evolve in an ERP SECO when the SECO vendor (orchestrator) is pushing an upgrade to a newer version. It is notable that the relations can be push-oriented, i.e., the orchestrator pushes a new version to the ISVs and VARs and eventually the customer, but also pull-oriented, i.e., the customer requests an older version from the ISVs/VARs end eventually the orchestrator. Jansen et al. (2012) referring to Popp (2010) numbers three distribution channels: (i) direct through VAR, (ii) indirect through

[9] Used in: van Angeren et al. (2011, 2011), Jansen et al. (2009b,a), Hilkert et al. (2010), Idu et al. (2011), van der Schuur et al. (2011).

[10] Used in: van Angeren et al. (2011), Burkard et al. (2012), Campbell and Ahmed (2010), Hanssen (2011), Jansen et al. (2009a, 2012), Kabbedijk and Jansen (2011), McGregor (2010), Pettersson et al. (2010), Riis and Schubert (2012), Viljainen and Kauppinen (2011), Idu et al. (2011), dos Santos and Werner (2011a,b), te Molder et al. (2011), van den Berk et al. (2010), van Ingen et al. (2011), van der Schuur et al. (2011).

[11] Used in: dos Santos and Werner (2011a,b), Burkard et al. (2012), Hilkert et al. (2010), Riis and Schubert (2012), van den Berk et al. (2010).

[12] Used in: Jansen et al. (2009a), Viljainen and Kauppinen (2011), van der Schuur et al. (2011)

[13] Used in: Campbell and Ahmed (2010).

[14] Used in: van Angeren et al. (2011).

[15] Used in: Jansen et al. (2009a), Viljainen and Kauppinen (2011, 2011), dos Santos and Werner (2011a,b), Burkard et al. (2012), Yu and Deng (2011), Kabbedijk and Jansen (2011), Riis and Schubert (2012), te Molder et al. (2011), van den Berk et al. (2010), van Ingen et al. (2011), van der Schuur et al. (2011).

[16] Used in: dos Santos and Werner (2011a,b), van den Berk et al. (2010).

[17] Used in: Jansen et al. (2009a, 2012), Viljainen and Kauppinen (2011, 2011), Bosch and Bosch-Sijtsema (2010c), Bosch (2009).

[18] Used in: Pettersson and Gil (2010), Hansen et al. (2011), Pettersson et al. (2010).

[19] Used in: Bosch (2010a, 2009, 2010b), Pettersson et al. (2010), dos Santos and Werner (2011a,b), Bosch and Bosch-Sijtsema (2010b,c,a), Jansen et al. (2012), Draxler and Stevens (2011), Kilamo et al. (2012), Viljainen and Kauppinen (2011), Scacchi and Alspaugh (2012), van Ingen et al. (2011), Weiss (2011).

[20] Used in: Anvaari and Jansen (2010), Bosch and Bosch-Sijtsema (2010b,c), Bosch (2009, 2010b), Campbell and Ahmed (2010), Dhungana et al. (2010), Hanssen (2011), Jansen et al. (2009a, 2012), Mizushima and Ikawa (2011), Seichter et al. (2010), Viljainen and Kauppinen (2011).

[21] Used in: Bosch and Bosch-Sijtsema (2010b,c).

[22] Used in: Bosch (2010b), Draxler and Stevens (2011).

[23] Used in: Campbell and Ahmed (2010).

[24] Used in: Jansen et al. (2009a).

[25] Used in: Viljainen and Kauppinen (2011).

[26] Used in: Jansen et al. (2009b,a, 2012), Bosch (2009), Boucharas et al. (2009), Draxler and Stevens (2011), Hilkert et al. (2010), Hunink et al. (2010), Riis and Schubert (2012), te Molder et al. (2011), van den Berk et al. (2010), Viljainen and Kauppinen (2011), Scacchi and Alspaugh (2012), Janner et al. (2008).

[27] Used in: Riis and Schubert (2012), Jansen et al. (2012, 2009b,a), Janner et al. (2008), Boucharas et al. (2009), Hanssen (2011), Popp (2011).

Table 6
The papers using existing SECOs.

SECO type	Papers	% of total
Proprietary	[45, 41, 2, 4, 10, 30, 54, 53, 16, 37, 44, 33, 6, 26, 87, 63, 82, 72, 62, 65]	22
FOSS	[6, 7, 48, 51, 50, 42, 31, 18, 57, 27, 89, 88, 73, 80, 76, 74, 68, 77, 75, 64, 70, 66, 60, 67, 83, 81, 85, 84]	31
No SECO	[40, 19, 1, 38, 39, 43, 3, 5, 9, 11, 8, 49, 59, 58, 52, 56, 13, 36, 46, 20, 23, 15, 22, 47, 21, 28, 35, 14, 29, 17, 12, 34, 55, 32, 24, 86, 71, 69, 61, 78, 79, 90]	47

service organization and (iii) direct to customer. Yu et al. (2008), Yu (2011) adopt the natural ecology types of symbiotic relationships to software symbiosis: mutualism, where both systems benefit from their relations, commensalism, where one system benefits from the relations while the other is unaffected, parasitism, where one system benefits and the other is harmed, amensalism, where one system is harmed and the other unaffected, competition, where both systems are harmed and neutralism where both systems are unaffected. Although, the symbiotic relations were described in the software symbiosis context rather than the social network, in our perspective, they could also be used to reflect SECO social network relations.

When looking into the niche player relationships, Kazman and Chen (2010) proposes the Metropolis model for the relationships between the actors in a SECO where it is consisted of the kernel that is responsible for platform and fundamental functionality, the periphery that is consisted of the prosumers building on top of the kernel's platform, and the masses that are the end-users. This can be parallelized to the "onion model" (Jergensen et al., 2011; Kilamo et al., 2012) appearing in FOSS projects, where the member involvement is similar to the layers of an onion: a member starts from the external layers having tasks with low responsibility, e.g., translation, and slowly moves to the inner layers gaining responsibilities. In another study of the developer behavior, Kabbedijk and Jansen (2011) studied the interaction of developers within the Ruby Github SECO and noted three different roles: the "lone wolf" that works mainly alone and produces big part of the system used by the rest of the users, the "networker" that is connected to several other developers and the "one day flies" that have created only one popular component without significant activity afterwards.

Communication among the different roles is also of interest. van der Schuur et al. (2011) study how knowledge is transferred within the different roles of a SECO while Fricker (2010) proposes the propagation of information in terms of requirements from the end-users or customers to the ecosystem with the requirement value chains.

4.5. Connection with industry

From the research questions that are mentioned in the beginning of this article, question 2.1 is investigating the use of real-world SECOs in the research. The purpose is to give a view on how close the connection of the research is to the industry. From the data collection process, we have compiled a list with all the papers that are using an existing SECO in their research as an object of study. Analyzing this list, we end up with the results that can be seen in Table 6. Going through the results, we notice that the slight majority of the papers (53%) is using an existing SECO in their research. The existing ecosystems are appearing in mainly two ways: (i) one or more SECOs are studied and the paper publishes study results, conclusions, interesting remarks as it is the case with Hanssen (2011) that describe the transition of a traditional waterfall-based software company to a SECO and (ii) a theory, framework, taxonomy or tool is developed based on literature, hypothesis or experience and then applied to one or more existing ecosystems to prove it, as it is the case in te Molder et al. (2011) where the Clopennes Assessment model is applied to an anonymized SECO to support the theory. In both of the cases, we argue that the use of existing ecosystems as objects of study increases the 'external' validity of the results.

Table 6 is separating the papers that study existing ecosystems in papers studying proprietary and free or open source software (FOSS) ecosystems. We separate the two kinds of ecosystems as they have significant differences. In a strict proprietary ecosystem, the source code and other artifacts produced are protected, as they are the products that would yield revenues to the ecosystem, while new actors would probably have to be certified in some way so they would be allowed to participate in the ecosystem. In a traditional FOSS ecosystem, the actors do not necessarily participate to obtain direct revenues from their activity in the ecosystem, while it is often much easier for an actor to participate in a FOSS than a proprietary SECO, since FOSS SECOs typically do no require any verification of new actors. Naturally, this simplistic way of separating proprietary and FOSS SECOs is only used to underline the differences of the two kinds of ecosystems. A majority of the SECOs would probably be categorized as a hybrid, combining elements from the two kinds. However, in the literature we note that papers studying FOSS SECOs are mostly concerned with problems of technical or social nature, while the papers studying proprietary SECOs include business and strategic problems. This is only natural, since FOSS projects allow the mining and processing of several details (like source code, commit logs, etc.) but they do not necessarily have a clear business model for the whole SECO or the participating actors (or at least it does not apear so in the literature). This underlines the importance of the research focusing on FOSS SECOs to include business and strategic perspectives. On the other hand, papers in the proprietary SECO group can get information about SECO strategies and positioning in the market, but it is harder to get access to proprietary information like source code, developer commits and so on.

Table 7 lists the existing SECOs used in the literature. The literature is studying 43 SECOs in total, out of which, 30 are studied in only one paper each. We note that out of the 12 SECOs studied in more than one paper (in this count we do not include the "Anonymized/not named" category), only two (GX Software and SAP) do not belong to the FOSS group and Eclipse being the most studied SECO (appearing in seven papers). Additionally, 18 out of the 43 studied SECOs are of proprietary nature. We explained this, by the additional challenge posed in gaining access to information in a proprietary SECOs in contradiction to a FOSS where data are usually accessed by mining a publicly available repository.

5. Discussion

The purpose of this study is to provide an overview of the field of software ecosystems by reviewing and analyzing the published literature. This work has been done based on the review protocol explained in Section 2.

In this work we did not include any evaluation of the quality of the relevant literature. The only consideration relating to the quality of a paper is the number of papers within the literature citing this paper, if any. It could be argued that a possible assessment of the quality of the literature could be undertaken to set focus on the gravity each paper should have in the analysis sections, e.g., 4.4.

Table 7
The SECOs appearing in the literature.

SECO name	Papers
Eclipse, Eclipse Foundation	6, 89, 73, 76, 67, 83, 85
GNOME	7, 51, 80, 74
Open Design Alliance	6, 76, 10, 16
Anonymized/not named	65, 82, 45
Brazilian Public Software (BPS)	88, 64, 33
Linux, Linux Kernel	50, 57, 70
Android	27, 66
GX Software	76, 10
Evince	7, 18
FOSS	42, 31
FreeBSD	50, 57
iPhone/iPad App Store	27, 72
SAP	53, 2
Apache Web Server	70
Artop	67
Brasero	7
CAS Software AG	37
CSoft	44
CubicEyes	6
Debian	60
Google Chrome	75
Google	
Gurux	2
Firefox	77
HIS GmbH	75
HISinOne	63
Mac App Store	26
Microsoft	72
Nokia Siemens Networks	2
Nautilus	87
Pharo	81
Ruby	68
S. Chand Edutech	84
SOOPS BV	62
Squeak	54
Symbian	68
TFN 200	67
UniImprove	41
Unity 30	75
US Department of Defense	30
WattDepot	48
WinMob	27
World of Worcraft	89

Apart from addressing the research questions and providing an overview in the field, we also identified several areas that are not covered in the literature body.

As already noted, the field of software ecosystems is not the only field inspired by the natural ecosystems. There has been significant amount of work done in other ecosystems like the business, social or natural ecosystems themselves. The SECO literature does not appear to examine work done in other ecosystems apart from a number of papers mentioned in Section 4. Possible intersections or parallelizations of the fields would allow the use of theories from the other fields or different perspectives in SECO problems.

An important ingredient of the success of an ecosystem is diversity. The differentiation of actors would allow niche creation. Statements similar to this have appeared several times in the literature. However, no concrete studies have been provided to prove a statement of this kind. Technical, organizational, business and social variability in harmonic symbiosis settings could bring more stability and possibly contribute to a healthier ecosystem.

The concept of health of an ecosystem, as explained in Section 4, section has been introduced to SECO from the business ecosystem theory. Measuring the health of an ecosystem would provide large benefits for the SECO industry and research. The health would provide indications on the future of the ecosystem and give possible feedback on applied changes in the ecosystem. However, apart from referring to SECO health, very few studies elaborate, analyze or measure the health of a software ecosystem.

The intellectual property rights and licensing issues are a focus point of a small part of the literature. Finding effective ways to address issues of this kind is of more importance than the attention it has been receiving in the literature. Issues of this kind are of importance both to the organizational perspectives of a SECO – how to organize the development in the ecosystem– but also in the business – how to develop the proper business/revenue models.

Quality assurance (QA) is a field that has also not been efficiently addressed in the literature. The adoption of traditional QA methods might not necessarily work in a SECO, because of the separation of platform and actors. Possibly, the proper QA strategies depend on the orchestration of the ecosystem and solutions might be specific to each SECO, however, there is a need for SECO specific QA strategies.

Finally, a field that has not been covered in the literature, is the organization of and decision making in SECOs. We recognize the high differentiation in the management models existing SECOs apply that would probably give reasons to why this field is not addressed in the literature. However, we argue that studies on that aspect of SECOs would assist, providing a more complete picture of the field.

6. Conclusion

Software ecosystems is an area that has been gaining in popularity the last five years. The software industry is moving towards software ecosystems, with platforms like Google Android and Apple iOS increasing in popularity, while research has increasing interest in the field, with the fourth year of a dedicated workshop (IWSECO 2012). This article is documenting a systematic literature review held on the field of software ecosystems. The purpose of this work was to provide an overview of the field and identify possible research issues or areas not covered. We found and analyzed 90 relevant papers from a gross total of 420 extracted from a list of scientific libraries. Based on this, we provided an overview of the definition of SECOs as it is defined in the literature including finding patterns in the different definitions provided and list the common main items that consist a SECO. We reported an increase in the research from 2007 to today. Additionally, we classified the research papers according to the result they reported and identified a lack in analytical models and an excess in report papers. Moreover, we defined "SECO architecture" and identified and analyzed the three main components that is consisted of: SECO Software Engineering, SECO Business and Management and SECO Relations. Finally, we examined the intersection of research and industry and found that half of the papers relate to the industry while at the same time most of them are focusing on FOSS SECOs. In conclusion, we identify the field of software ecosystems as a new field of growing importance and potential both in research and industry.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments that greatly improved the quality of this paper.

This work has been partially funded by the Net4Care project within Caretech Innovation (http://www.caretechinnovation.dk/projekter/net4care/).

Appendix A. Literature body

1. Capuruço and Capretz (2010)
2. Popp (2011)
3. Yu and Deng (2011)

4. te Molder et al. (2011)
5. dos Santos and Werner (2011b)
6. van Angeren et al. (2011)
7. Mens and Goeminne (2011)
8. Barbosa and Alves (2011)
9. Alspaugh et al. (2009)
10. Jansen et al. (2009a)
11. Fricker (2009)
12. Campbell and Ahmed (2010)
13. Seichter et al. (2010)
14. Dhungana et al. (2010)
15. Lungu et al. (2010b)
16. van den Berk et al. (2010)
17. Cataldo and Herbsleb (2010)
18. Goeminne and Mens (2010)
19. Bosch (2010a)
19. Bosch (2009)
21. Kazman and Chen (2010)
22. Lungu and Lanza (2010)
23. Pettersson et al. (2010)
24. Scacchi (2010b)
25. Boucharas et al. (2009)
26. Brummermann et al. (2011)
27. Anvaari and Jansen (2010)
28. Hunink et al. (2010)
29. dos Santos and Werner (2010)
30. McGregor (2010)
31. Scacchi (2007a)
32. Krishna and Srinivasa (2011)
33. Alves and Pessôa (2010)
34. Briscoe (2010)
35. Fricker (2010)
36. Scacchi (2010a)
37. Hilkert et al. (2010)
38. Bosch and Bosch-Sijtsema (2010c)
39. Bosch and Bosch-Sijtsema (2010a)
40. Kazman et al. (2012)
41. Schneider et al. (2010)
42. Yu and Woodard (2009)
43. Bosch (2010b)
44. Hanssen (2011)
45. Bosch and Bosch-Sijtsema (2010b)
46. Scacchi (2007b)
47. Lungu et al. (2010a)
48. Brewer and Johnson (2010)
49. Pettersson and Gil (2010)
50. Yu et al. (2008)
51. Lungu et al. (2009)
52. An (2009)
53. Janner et al. (2008)
54. Lungu (2008)
55. Hindle et al. (2010)
56. Jansen et al. (2009b)
57. Yu et al. (2007)
58. Schugerl et al. (2009)
59. Kakola (2010)
60. Ververs et al. (2011)
61. van Angeren et al. (2011)
62. Scholten et al. (2012)
63. Brummermann et al. (2012)
64. Stefanuto et al. (2011)
65. van der Schuur et al. (2011)
66. van Ingen et al. (2011)
67. Weiss (2011)
68. Robbes and Lungu (2011)
69. Schmerl et al. (2011)
70. Yu (2011)
71. dos Santos and Werner (2011a)
72. Idu et al. (2011)
73. Draxler et al. (2011a)
74. Jergensen et al. (2011)
75. Scacchi and Alspaugh (2012)
76. Jansen et al. (2012)
77. Kilamo et al. (2012)
78. Pettersson and Vogel (2012)
79. Kajan et al. (2011)
80. Pérez et al. (2012)
81. Neu et al. (2011)
82. Riis and Schubert (2012)
83. Mizushima and Ikawa (2011)
84. Kabbedijk and Jansen (2011)
85. Draxler and Stevens (2011)
86. Burkard et al. (2012)
87. Viljainen and Kauppinen (2011)
88. Alves et al. (2011)
89. Draxler et al. (2011b)
90. Widjaja and Buxmann (2011)

References

Alspaugh, T., Asuncion, H., Scacchi, W., 2009. The role of software licenses in open architecture ecosystems. In: First International Workshop on Software Ecosystems (IWSECO-2009), Citeseer, pp. 4–18.

Alves, A.M., Pessôa, M., 2010. Brazilian public software: beyond sharing. In: in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 73–80.

Alves, A.M., Pessoa, M., Salviano, C.F., 2011. Towards a systemic maturity model for public software ecosystems. In: O'Connor, R.V., Rout, T., McCaffery, F., Dorling, A. (Eds.), Software Process Improvement and Capability Determination, vol. 155 of Communications in Computer and Information Science. Springer, Berlin/Heidelberg, pp. 145–156, 10.1007/978-3-642-21233-8_13.

An, H., 2009. Research on software problems based on ecological angle. In: International Conference on Environmental Science and Information Application Technology 2009 (ESIAT 2009), pp. 11–14.

van Angeren, J., Blijleven, V., Jansen, S., 2011. Relationship intimacy in software ecosystems: a survey of the dutch software industry. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 68–75.

van Angeren, J., Kabbedijk, J., Popp, K.M., 2011. A survey of associate models used within large software ecosystems. In: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, pp. 27–39.

Anvaari, M., Jansen, S., 2010. Evaluating architectural openness in mobile software platforms. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 85–92.

Barbosa, O., Alves, C., 2011. A systematic mapping study on software ecosystems. In: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, pp. 15–26.

Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice. Addison-Wesley Longman Publishing Co., Inc., Boston.

van den Berk, I., Jansen, S., Luinenburg, L., 2010. Software ecosystems: a software ecosystem strategy assessment model. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 127–134.

Bosch, J., 2009. From software product lines to software ecosystems. In: Proceedings of the 13th International Software Product Line Conference, Carnegie Mellon University, Pittsburgh, PA, USA, pp. 111–119.

Bosch, J., 2010a. Architecture challenges for software ecosystems. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 93–95.

Bosch, J., 2010b. Architecture in the age of compositionality. In: Babar, M., Gorton, I. (Eds.), Software Architecture, volume 6285 of Lecture Notes in Computer Science. Springer Berlin/Heidelberg, pp. 1–4, http://dx.doi.org/10.1007/978-3-642-15114-9_1

Bosch, J., Bosch-Sijtsema, P., 2010a. Coordination between global agile teams: from process to architecture. In: Smite, D., Moe, N.B., Agerfalk, P.J. (Eds.), Agility Across Time and Space. Springer, Berlin/Heidelberg, pp. 217–233, http://dx.doi.org/10.1007/978-3-642-12442-6_15

Bosch, J., Bosch-Sijtsema, P., 2010b. From integration to composition. On the impact of software product lines global development and ecosystems. Journal of Systems and Software 83, 67–76, SI: Top Scholars.

Bosch, J., Bosch-Sijtsema, P.M., 2010c. Softwares product lines, global development and ecosystems: Collaboration in software engineering. In: Mistrik, I., van der Hoek, A., Grundy, J., Whitehead, J. (Eds.), Collaborative Software Engineering.

Springer, Berlin/Heidelberg, pp. 77–92, http://dx.doi.org/10.1007/978-3-642-10294-3_4

Boucharas, V., Jansen, S., Brinkkemper, S., 2009. Formalizing software ecosystem modeling. In: in: Proceedings of the 1st International Workshop on Open Component Ecosystems, ACM, New York, NY, USA, pp. 41–50.

Brewer, R., Johnson, P., 2010. Wattdepot: an open source software ecosystem for enterprise-scale energy data collection storage analysis and visualization. In: in: First IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 91–95.

Briscoe, G., 2010. Complex adaptive digital ecosystems. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 39–46.

Brummermann, H., Keunecke, M., Schmid, K., 2011. Variability issues in the evolution of information system ecosystems. In: Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems, ACM, New York, NY, USA, pp. 159–164.

Brummermann, H., Keunecke, M., Schmid, K., 2012. Formalizing distributed evolution of variability in information system ecosystems. In: Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, ACM, New York, NY, USA, pp. 11–19.

Burkard, C., Widjaja, T., Buxmann, P., 2012. Software ecosystems. Business and Information Systems Engineering 4, 41–44, http://dx.doi.org/10.1007/s12599-011-0199-8.

Campbell, P.R.J., Ahmed, F., 2010. A three-dimensional view of software ecosystems. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 81–84.

Capuruço, R.A.C., Capretz, L.F., 2010. Integrating recommender information in social ecosystems decisions. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 143–150.

Cataldo, M., Herbsleb, J.D., 2010. Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 65–72.

Dhungana, D., Groher, I., Schludermann, E., Biffl, S., 2010. Software ecosystems vs natural ecosystems learning from the ingenious mind of nature. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 96–102.

Draxler, S., Jung, A., Boden, A., Stevens, G., 2011a. Workplace warriors: identifying team practices of appropriation in software ecosystems. In: in: Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering, ACM, New York, NY, USA, pp. 57–60.

Draxler, S., Jung, A., Stevens, G., 2011b. Managing software portfolios: a comparative study. In: Costabile, M., Dittrich, Y., Fischer, G., Piccinno, A. (Eds.), End-User Development, vol. 6654 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 337–342, http://dx.doi.org/10.1007/978-3-642-21530-8_36

Draxler, S., Stevens, G., 2011. Supporting the collaborative appropriation of an open software ecosystem. Computer Supported Cooperative Work (CSCW) 20, 403–448, http://dx.doi.org/10.1007/s10606-011-9148-9

Fricker, S., 2009. Specification and analysis of requirements negotiation strategy in software ecosystems. In: in: First International Workshop on Software Ecosystems (IWSECO-2009), Citeseer, pp. 19–33.

Fricker, S., 2010. Requirements value chains: stakeholder management and requirements engineering in software ecosystems. In: Wieringa, R., Persson, A. (Eds.), Requirements Engineering: Foundation for Software Quality, vol. 6182 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 60–66, http://dx.doi.org/10.1007/978-3-642-14192-8_7

Goeminne, M., Mens, T., 2010. A framework for analysing and visualising open source software ecosystems. In: in: Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), ACM, New York, NY, USA, pp. 42–47.

Hansen, K.M., Jonasson, K., Neukirchen, H., 2011. Controversy corner. An empirical study of software architectures' effect on product quality. Journal of Systems and Software 84, 1233–1243.

Hanssen, G.K., 2011. A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. Journal of Systems and Software, http://dx.doi.org/10.1016/j.jss.2011.04.020

den Hartigh, E., Tol, M., Visscher, W., 2006. The health measurement of a business ecosystem. In: in: Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting.

Hilkert, D., Wolf, C.M., Benlian, A., Hess, T., 2010. The "as-a-service"-paradigm and its implications for the software industry – insights from a comparative case study in crm software ecosystems. In: Aalst, W., Mylopoulos, J., Sadeh, N.M., Shaw, M.J., Szyperski, C., Tyrvainen, P., Jansen, S., Cusumano, M.A. (Eds.), Software Business, vol. 51 of Lecture Notes in Business Information Processing. Springer, Berlin/Heidelberg, pp. 125–137, http://dx.doi.org/10.1007/978-3-642-13633-7_11

Hindle, A., Herraiz, I., Shihab, E., Jiang, Z.M., 2010. Mining challenge 2010: Freebsd gnome desktop and debian/ubuntu. In: in: 7th IEEE Working Conference on Mining Software Repositories (MSR), pp. 82–85.

Hunink, I., van Erk, R., Jansen, S., Brinkkemper, S., 2010. Industry taxonomy engineering: the case of the european software ecosystem. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 111–118.

Iansiti, M., Levien, R., 2004a. The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy Innovation and Sustainability. Harvard Business Press, Boston.

Iansiti, M., Levien, R., 2004b. Strategy as ecology. Harvard Business Review 82, 68–81.

Idu, A., van de Zande, T., Jansen, S., 2011. Multi-homing in the apple ecosystem: why and how developers target multiple apple app stores. In: in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 122–128.

van Ingen, K., van Ommen, J., Jansen, S., 2011. Improving activity in communities of practice through software release management. In: in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 94–98.

Janner, T., Schroth, C., Schmid, B., 2008. Modelling service systems for collaborative innovation in the enterprise software industry – the st. gallen media reference model applied. In: in: IEEE International Conference on Services Computing 2008 (SCC'08), pp. 145–152.

Jansen, S., Brinkkemper, S., Finkelstein, A., 2009a. Business network management as a survival strategy: a tale of two software ecosystems. In: in: First International Workshop on Software Ecosystems (IWSECO-2009), Citeseer, pp. 34–48.

Jansen, S., Brinkkemper, S., Souer, J., Luinenburg, L., 2012. Shades of gray: opening up a software producing organization with the open software enterprise model. Journal of Systems and Software 85, 1495–1510, Software Ecosystems.

Jansen, S., Finkelstein, A., Brinkkemper, S., 2009b. A sense of community: A research agenda for software ecosystems. In: in: 31st International Conference on Software Engineering – Companion Volume, 2009. ICSE-Companion 2009, pp. 187–190.

Jergensen, C., Sarma, A., Wagstrom, P., 2011. The onion patch: migration in open source ecosystems. In: in: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ACM, New York, NY, USA, pp. 70–80.

Kabbedijk, J., Jansen, S., 2011. Steering insight: An exploration of the ruby software ecosystem. In: Regnell, B., Weerd, I., Troyer, O., Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M.J., Szyperski, C. (Eds.), Software Business, vol. 80 of Lecture Notes in Business Information Processing. Springer, Berlin/Heidelberg, pp. 44–55, http://dx.doi.org/10.1007/978-3-642-21544-5_5

Kajan, E., Lazic, L., Maamar, Z., 2011. Software engineering framework for digital service-oriented ecosystem. In: in: 19th Telecommunications Forum (TELFOR), pp. 1320–1323.

Kakola, T., 2010. Standards initiatives for software product line engineering and management within the international organization for standardization. In: in: 43rd Hawaii International Conference on System Sciences (HICSS), 2010, pp. 1–10.

Kazman, R., Chen, H.M., 2010. The metropolis model and its implications for the engineering of software ecosystems. In: in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, ACM, New York, NY, USA, pp. 187–190.

Kazman, R., Gagliardi, M., Wood, W., 2012. Scaling up software architecture analysis. Journal of Systems and Software 85, 1511–1519, Software Ecosystems.

Kilamo, T., Hammouda, I., Mikkonen, T., Aaltonen, T., 2012. From proprietary to open source-growing an open source ecosystem. Journal of Systems and Software 85, 1467–1478.

Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Engineering 2.

Krishna, R.P.M., Srinivasa, K.G., 2011. Analysis of projects and volunteer participation in large scale free and open source software ecosystem. SIGSOFT Software Engineering Notes 36, 1–5.

Lungu, M., 2008. Towards reverse engineering software ecosystems. In: in: IEEE International Conference on Software Maintenance, ICSM 2008, pp. 428–431.

Lungu, M., Lanza, M., 2010. The small project observatory: a tool for reverse engineering software ecosystems. In: in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ACM, New York, NY, USA, pp. 289–292.

Lungu, M., Lanza, M., Gîrba, T., Robbes, R., 2010a. The small project observatory: visualizing software ecosystems. Science of Computer Programming 75, 264–275, Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008).

Lungu, M., Malnati, J., Lanza, M., 2009. Visualizing gnome with the small project observatory. In: in: Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories, 2009 (MSR'09), pp. 103–106.

Lungu, M., Robbes, R., Lanza, M., 2010b. Recovering inter-project dependencies in software ecosystems. In: in: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, ACM, New York, NY, USA, pp. 309–312.

McGregor, J.D., 2010. A method for analyzing software product line ecosystems. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 73–80.

Mens, T., Goeminne, M., 2011. Analysing the evolution of social aspects of open source software ecosystems. In: in: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, pp. 1–14.

Messerschmitt, D., Szyperski, C., 2005. Software Ecosystem: Understanding An Indispensable Technology and Industry. MIT Press Books 1, London, England.

Mizushima, K., Ikawa, Y., 2011. A structure of co-creation in an open source software ecosystem: a case study of the eclipse community. In: in: 2011 Proceedings of

PICMET'11, Technology Management in the Energy Smart World (PICMET), pp. 1–8.
te Molder, J., van Lier, B., Jansen, S., 2011. Clopenness of systems: The interwoven nature of ecosystems. In: in: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, pp. 52–64.
Neu, S., Lanza, M., Hattori, L., D'Ambros, M., 2011. Telling stories about gnome with complicity. In: in: 2011 6th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT), pp. 1–8.
Pettersson, O., Gil, D., 2010. On the issue of reusability and adaptability in m-learning systems. In: in: 2010 6th IEEE International Conference on Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE), pp. 161–165.
Pettersson, O., Svensson, M., Gil, D., Andersson, J., Milrad, M., 2010. On the role of software process modeling in software ecosystem design. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 103–110.
Pettersson, O., Vogel, B., 2012. Reusability and interoperability in mobile learning: a study of current practices. In: in: 2012 IEEE Seventh International Conference onWireless, Mobile and Ubiquitous Technology in Education (WMUTE), pp. 306–310.
Pérez, J., Deshayes, R., Goeminne, M., Mens, T., 2012. Seconda: software ecosystem analysis dashboard. In: in: 16th European Conference on Software Maintenance and Reengineering (CSMR), pp. 527–530.
Popp, K., 2010. Goals of software vendors for partner ecosystems – a practitioner's view. Software Business, 181–186.
Popp, K.M., 2011. Hybrid revenue models of software companies and their relationship to hybrid business models. In: in: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, pp. 77–88.
Riis, P., Schubert, P., 2012. Upgrading to a new version of an erp system: a multilevel analysis of influencing factors in a software ecosystem. In: in: 45th Hawaii International Conference on System Science (HICSS), pp. 4709–4718.
Robbes, R., Lungu, M., 2011. A study of ripple effects in software ecosystems (nier track). In: in: Proceedings of the 33rd International Conference on Software Engineering, ACM, New York, NY, USA, pp. 904–907.
dos Santos, R.P., Werner, C., 2011a. Treating business dimension in software ecosystems. In: in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 197–201.
dos Santos, R.P., Werner, C.M.L., 2010. Revisiting the concept of components in software engineering from a software ecosystem perspective. In: in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 135–142.
dos Santos, R.P., Werner, C.M.L., 2011b. A proposal for software ecosystem engineering. In: in: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, pp. 40–51.
Scacchi, W., 2007a. Free/open source software development: recent research results and emerging opportunities. In: in: The 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: Companion Papers, ACM, New York, NY, USA, pp. 459–468.
Scacchi, W., 2007b. Free/open source software development: recent research results and methods. In: Zelkowitz, M.V. (Ed.), Architectural Issues, Elsevier, vol. 69 of Advances in Computers. , pp. 243–295.
Scacchi, W., 2010a. Collaboration practices and affordances in free/open source software development. In: Mistrík, I., van der Hoek, A., Grundy, J., Whitehead, J. (Eds.), Collaborative Software Engineering. Springer, Berlin/Heidelberg, pp. 307–327, http://dx.doi.org/10.1007/978-3-642-10294-3_15
Scacchi, W., 2010b. The future of research in free/open source software development. In: in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, ACM, New York, NY, USA, pp. 315–320.
Scacchi, W., Alspaugh, T.A., 2012. Understanding the role of licenses and evolution in open architecture software ecosystems. Journal of Systems and Software 85, 1479–1494.
Schmerl, B., Garlan, D., Dwivedi, V., Bigrigg, M.W., Carley, K.M., 2011. Sorascs: a case study in soa-based platform design for socio-cultural analysis. In: Proceedings of the 33rd International Conference on Software Engineering, ACM, New York, NY, USA, pp. 643–652.
Schneider, K., Meyer, S., Peters, M., Schliephacke, F., Mörschbach, J., Aguirre, L., 2010. Feedback in context: supporting the evolution of it-ecosystems. In: Ali Babar, M., Vierimaa, M., Oivo, M. (Eds.), Product-Focused Software Process Improvement, vol. 6156 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 191–205, 10.1007/978-3-642-13792-1_16.
Scholten, U., Fischer, R., Zirpins, C., 2012. The dynamic network notation: harnessing network effects in paas-ecosystems. In: Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners, ACM, New York, NY, USA, pp. 25–30.
Schugerl, P., Rilling, J., Witte, R., Charland, P., 2009. A quality perspective of software evolvability using semantic analysis. In: IEEE International Conference on Semantic Computing, ICSC'09, pp. 420–427.
van der Schuur, H., Jansen, S., Brinkkemper, S., 2011. The power of propagation: on the role of software operation knowledge within software ecosystems. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 76–84.
Seichter, D., Dhungana, D., Pleuss, A., Hauptmann, B., 2010. Knowledge management in software ecosystems: software artefacts as first-class citizens. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ACM, New York, NY, USA, pp. 119–126.
Shaw, M., 2003. Writing good software engineering research papers. In: Minitutorial for Proc ICSE"03.
Stefanuto, G., Spiess, M., Alves, A.M., Castro, P.F.D., 2011. Quality in software digital ecosystems the users perceptions. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 85–88.
Ververs, E., van Bommel, R., Jansen, S., 2011. Influences on developer participation in the debian software ecosystem. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, ACM, New York, NY, USA, pp. 89–93.
Viljainen, M., Kauppinen, M., 2011. Software ecosystems: a set of management practices for platform integrators in the telecom industry. In: Regnell, B., Weerd, I., Troyer, O., Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M.J., Szyperski, C. (Eds.), Software Business, vol. 80 of Lecture Notes in Business Information Processing. Springer, Berlin/Heidelberg, pp. 32–43, 10.1007/978-3-642-21544-5_4.
Weiss, M., 2011. Economics of collectives. In: Proceedings of the 15th International Software Product Line Conference, vol. 2, ACM, New York, NY, USA, pp. 39:1–39:8.
Widjaja, T., Buxmann, P., 2011. Compatibility of software platforms. In: Heinzl, A., Buxmann, P., Wendt, O., Niche Weitzel, T. (Eds.), Theory-Guided Modeling and Empiricism in Information Systems Research. Physica-Verlag HD, Berlin Heidelberg, Germany, pp. 15–41, http://dx.doi.org/10.1007/978-3-7908-2781-1_2
Yu, E., Deng, S., 2011. Understanding software ecosystems: A strategic modeling approach. In: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, pp. 65–76.
Yu, L., 2011. Coevolution of information ecosystems: a study of the statistical relations among the growth rates of hardware, system software, and application software. SIGSOFT Software Engineering Notes 36, 1–5.
Yu, L., Ramaswamy, S., Bush, J., 2007. Software evolvability: an ecosystem point of view. In: in: Third International IEEE Workshop on Software Evolvability, pp. 75–80.
Yu, L., Ramaswamy, S., Bush, J., 2008. Symbiosis and software evolvability. IT Professional 10, 56–62.
Yu, S., Woodard, C., 2009. Innovation in the programmable web: characterizing the mashup ecosystem. In: Feuerlicht, G., Lamersdorf, W. (Eds.), Service-Oriented Computing – ICSOC 2008 Workshops, vol. 5472 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 136–147, http://dx.doi.org/10.1007/978-3-642-01247-1_13.

Konstantinos Manikas is a PhD scholar at the Department of Computer Science of Copenhagen University. His main research areas are software architecture and software ecosystems with interest in telemedicince and healthcare IT.

Klaus Marius Hansen is a professor of Software Development at the University of Copenhagen. He received a Ph.D. degree in Computer Science from Aarhus University in 2002 and his research focuses on software technology and use in particular in relation to pervasive and dependable computing.

Paper 2

Reviewing the Health of Software Ecosystems – A Conceptual Framework Proposal

Manikas, K. and Hansen, K. M. (2013b). Reviewing the health of software ecosystems - a conceptual framework proposal. In Alves, C. F., Hanssen, G. K., Bosch, J., and Jansen, S., editors, *Proceedings of the 5th International Workshop on Software Ecosystems, Potsdam, Germany, June 11, 2013*, volume 987, pages 33–44

Reviewing the Health of Software Ecosystems – A Conceptual Framework Proposal

Konstantinos Manikas and Klaus Marius Hansen

Department of Computer Science (DIKU)
University of Copenhagen
Njalsgade 128
2300 Copenhagen S
Denmark
{kmanikas,klausmh}@diku.dk

Abstract. The health of a software ecosystem is an indication of how well the ecosystem is functioning. The measurement of health can point to issues that need to be addressed in the ecosystem and areas for the ecosystem to improve. However, the software ecosystem field lacks an applicable way to measure and evaluate health. In this work, we review the literature related to the concept of software ecosystem health and the literature that inspired the software ecosystem health literature (a total of 23 papers) and (i) identify that the main source of inspiration is the health of business ecosystems while also influenced by theories from natural ecosystems and open source, (ii) identify two areas where software ecosystems differ from business and natural ecosystems, and (iii) propose a conceptual framework for defining and measuring the health of software ecosystems.

Key words: software ecosystems, ecosystem health, software ecosystem health framework, software ecosystem health measurement

1 Introduction

The notion of software ecosystems (SECOs) is gaining popularity as a means of expanding development, better positioning in the market, or increasing revenues. There is a number of definitions of SECOs in the literature [1, 2, 3, 4]. In this work we define a software ecosystem as *"the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services. Each actor is motivated by a set of interests or business models and connected to the rest of the actors and the ecosystem as a whole with symbiotic relationships, while, the technological platform is structured in a way that allows the involvement and contribution of the different actors"* [5]. Today, software ecosystems come with a wide variability of characteristics: platform structure, actor participation, ecosystem orchestration, and revenue models, to name a few. This makes the establishment of methods for measuring and evaluating the activity of the ecosystem challenging. The SECO literature refers to the concept of "health" of an ecosystem as a way to monitor ecosystem activity, identify and

predict areas for improvement, and evaluate changes in the ecosystem. However, the measurement of the SECO health is not yet fully achieved.

We tentatively define the health of a software ecosystem as the ability of the ecosystem to endure and remain variable and productive over time. In this work, we aim to get closer to SECO health measurement by reviewing the literature that is elaborating on SECO health. In doing so, we identify that the SECO health literature is borrowing definitions and measurement of health from other fields and expand our literature review focus to include additional ecosystem health fields (explained in section 2). We review the wider ecosystem health literature body and report the health definitions and measurements (section 3). We identify two main differences between SECOs and business and natural ecosystems and, based on previous work, we propose a conceptual framework for defining and measuring the health of SECOs (section 4). Finally, in section 5 we discuss threats to validity and future work and conclude in section 6.

2 Defining the Health Literature Body

The method used for defining the literature body consisted of the following steps:

(i) Defining the SECO health literature. To define the literature related to the SECO health, we used as input the papers identified in our recent systematic literature review [5]. In [5], we identified a number of papers referring to the concept of ecosystem health. The papers have a wide variability on the level of detail they provide on the ecosystem health ranging from mere reference to the concept (e.g., [6, 7, 8]) to papers in which health forms part of the main focus (e.g., [9, 10]).

(ii) Defining wider ecosystem health literature. While examining the SECO health literature, we noticed that the definition and analysis of health is borrowed from other types of ecosystems not covered by the SECO health literature . Using the "snowballing technique" [11], we followed the references of the SECO health literature and evaluated whether these are related to the health of an ecosystem [1]. The criteria for accepting a paper in the literature was that it would (a) define the health or sustainability of an ecosystem or (b) elaborate on ways of measuring health.

Table 1 shows the literature body and the papers that are referenced by each document. The SECO health literature that resulted from step (i) is the first row while the literature from step (ii) are the remaining. We have organized the papers into categories according to their field: software ecosystems (SECOs), business ecosystems (BECOs)[2], natural ecosystems, and open source software (OSS). The main purpose of listing the categories is to show the fields that

[1] We also followed references of the selected references that appeared relevant, resulting in a number of papers ([12, 13])

[2] Paper [27] is defining the field of "IT ecosystems", though, as a BECO with IT products.

Type	Paper	Source
SECO	[14, 10, 15, 9, 16, 17, 7, 18, 8, 19, 20, 21, 6]	Health literature from [5]
BECO	[22]	[14, 9, 21, 10, 15, 23, 24]
	[25]	[14, 7, 10, 16, 6, 23]
	[23]	[14, 10, 9, 16, 18]
	[26]	[18, 8, 23]
	[27]	[20]
	[24]	[9]
Natural ecosystems	[28]	[9]
	[12]	[28]
	[13]	[28]
OSS	[29]	[19]
Total:	23	

Table 1. List of the documents in the health literature and the documents referring to them.

influenced SECO health. In this work, we have not looked at the health definition and measurement in the other fields outside the references of the SECO health literature and, thus, do not claim that these papers are representative of each field.

3 Ecosystem Health

Following the separation of ecosystem fields in Table 1, we list and discuss the papers per ecosystem that have influenced the SECO health literature.

3.1 Natural Ecosystems

The field of (natural) ecosystems inspired the rest of the ecosystem fields examined here (BECO and SECO) and it is the field where the concept of ecosystem health was initially formulated. Costanza [12], defines a healthy ecosystem as "being 'stable and sustainable'; maintaining its organization and autonomy over time and its resilience to stress". In addition, Rapport et al. [28], referring to a collection of papers in the literature, define three indicators for health of an ecosystem: *Vigor* that indicates how active or productive an ecosystem is, *Organization* that indicates the variability of species, and *Resilience* that indicates the ability of the ecosystem to "maintain structure and function in the presence of stress". The characterization of an ecosystem in terms of structure and function is also discussed by Schaeffer et al. in [13]. They parallelize ecosystem health

with human health and define it as the "absence of disease". They identify structure as "numbers of kinds of organisms, biomass etc." and function as "activity, production, decomposition etc.". These are seen as measures used to define the ecosystem health. Furthermore, Schaeffer et al., referring to the literature, list four ways that structure and function may be connected: (a) tightly connected, where neither can change without the change of the other, (b) structure changes does not affect function, (c) function changes do not affect structure, and, (d) structure and function appear unconnected.

3.2 Business Ecosystems

BECO health is the area that has inspired most of the SECO health literature. In the BECO literature, the concept of health is mainly defined as the ability of a BECO to provide "durably growing opportunities for its members and for those who depend on it" [26][3]. Iansiti and Levien [25, 26, 22] and Iansiti and Richards [27] define the health of a business ecosystem using three measures:

Productivity. Inspired by natural ecosystems' ability to create energy from input sources (e.g., sunlight or mineral nutrients), BECO productivity is the ability of an ecosystem to "convert raw materials of innovation into lowered costs and new products and functions" [26]. Productivity in BECOs can be measured by means of (a) total factor productivity, (b) productivity improvement over time, and (c) delivery of innovations, the ability of the ecosystem to adapt and deliver to its members new technologies, ideas, or process.

Robustness. The ability of the ecosystem to sustain shocks, perturbations, and disruptions. Robustness is measured in terms of (a) survival rates, the survival of actors over time, (b) persistence of ecosystem structure, the extent to which actor relationships are kept unchanged, (c) predictability, the extent to which even if shocks alter the relationships of actors, a main core of the ecosystem remains solid, (d) limited obsolescence, whether the ecosystem has a limited invested technology or components that becomes obsolete after a shock, and (e) continuity of use experience and use cases, the extent to which products gradually evolve in response to new technologies rather than changing abruptly.

Niche Creation or Innovation. The ability of the ecosystem to increase meaningful actor diversity over time. Niche creation is measured in terms of (i) growth in company variety and (ii) growth in product and technical variety (value creation) that measures the increase in value the growth brings.

Iansiti and Levien and Iansiti and Richards also propose three ecosystem actor roles, inspired by natural ecosystems, that affect the health of a BECO:

Keystone. Is an actor that normally occupies or creates highly connected hubs of actors and promotes the health of the ecosystem by providing value to

[3] Similar definitions appear in [22, 27]

the surrounding actors. Keystones promote the health of the ecosystem by increasing the variability, provide value to the connected actors and thus increase productivity, and increase robustness by protecting connected actors from external shocks.

Dominators. Are the actors that control the "value capture and value creation" [22] of the ecosystem. They tend to expand by taking over the functions of other actors thus eventually eliminating the actors. Dominators are harmful for the health of an ecosystem as they reduce diversity

Niche (players or firms). Usually form the main volume of the ecosystem actors drawing value from the keystones. A niche player aims to separate from the other niche players by developing special functions.

Typically, a keystone provides value to a number of actors that can be either niche players trying to develop or dominators trying to dominate the functions of the surrounding actors. The roles of the BECO actors are also examined by Iyer and Lee [24]. They classify the actors in an ecosystem in (a) hubs, (b) brokers that connect two sets of actors, and, (c) bridges that are essential for the connectedness of the ecosystem. A hub can demonstrate keystone, dominator, or niche player characteristics.

Hartigh et al. [23] use the work of Iansiti and Levien and Iansiti and Richards (referred to as "Iansiti" hereafter) to measure the health of the Dutch IT industry. They define BECO health using two long-term parameters: the financial well-being and strength of the network and break down the health in two components: partner health and network health. Partner health evaluates the health of each individual actor of the ecosystem. A healthy ecosystem is composed of productive actors contributing to the productivity of the ecosystem while unproductive actors will have difficulty surviving. The survival of the actors is analogous to the Iansiti robustness measure. Network health is measured in terms of actor connectivity. Highly connected actors contribute to the robustness of the ecosystem as the actors are not easily affected by external shocks. In addition, a healthy ecosystem contains clusters of different nature, thus increasing the possibility of niche creation.

3.3 OSS

Wahyudin et al. [29] study the concept of health in OSS projects. They define the health of an OSS project as "survivability", the ability of the project to survive throughout time. An OSS project is healthy and survives if the software produced by the project is used by a number of users and maintained by a number of developers. They identify three measures that affect the health of an open source project:

The developer community liveliness. The project should attract new developers and keep the existing by boosting their motivation. Wahyudin et al. break down an OSS developer's motivation in intellectual stimulation, skill enhancement, and access to source code and user needs.

The user community liveliness. The users of OSS software play an active role in the evolution of the project by reporting bugs and requesting new features. A large, active user community indicates that the software produced is usable and of good quality.

The product quality. A product that is competitive with commercial products in use and quality will attract users and developers, increase the activity in the project, and therefore enhance survivability.

3.4 Software Ecosystems

In the field of SECO, Berk et al. [9] propose SECO-SAM, a model for the assessment of a SECO strategy based on SECO health. In their model, they make an analogy between the health of an ecosystem and human health and propose that SECO health is influenced by the biology of the ecosystem, the lifestyle, the environment, and the intervention of healthcare organizations while they measure the SECO health adopting the Iansiti productivity, robustness, and niche creation (PRN) measures. Jansen et al. [10] elaborate on a three-level model of SECOs, published in [7], consisting of the organization scope level, SECO level, and software supply network level. They define SECO health as a characteristic of the software supply network level using the Iansiti PRN measures. Additionally, they propose the application of the Hartigh et al. [23] measures for defining the health at the SECO level. Angeren et al. [14] show that SECO robustness of the Iansiti PRN measures is an important factor for vendors that choose to depend on a SECO.

In OSS, McGregor [20] translates the Iansiti PRN measures to measures that can be applied to open source projects, while Kilamo et al. [18] propose a framework for going from a proprietary to a Free/Libre/Open Source Software (FLOSS) SECO. One of the framework activities is setting up a "community watchdog" that assesses the community, the software, and "how well the objectives of the company are met". The watchdog indirectly assesses the health while they provide a number of measures to be applied in FLOSS SECOs.

Looking at the SECO health literature, we note that the main source of inspiration is BECO health when trying to define and measure SECO health or health-related parameters (e.g., keystone-dominator strategies) with 11 out of the 13 papers referring to at least one of the Iansiti authored papers [22, 25, 26, 27]. Although the health of a BECO is very similar to the SECO health, we identify a number of differences between the two. In the next section, we explain the differences and build on top of the existing literature to define a framework for SECO health.

4 A SECO Health Proposal

When analyzing the health of SECOs, we identify that similarly to BECOs and natural ecosystems, the set of actors, their activity and the network they form

is an indication of the level of prosperity and sustainability of the ecosystem. However, one main difference of SECOs from BECOs and natural ecosystems is in the nature of the products of the actors and, eventually, of the whole ecosystem. The BECO approach explained in the previous section is aligned with the natural ecosystem approach of actors and products, where the products of the ecosystem (i.e. energy) are represented by the actors (i.e., species) by enclosing energy in the energy flow between the species. In other words, a herbivorous species eats a plant and is eaten by a carnivorous species. This herbivorous species is both an actor and a product in the ecosystem and changes in the health of this species (e.g., number decrease) affects the energy enclosed (product) by this species and, thus, the carnivorous species.

In SECOs, the actors are differentiated from the products. The main product of the actors is software, either as a common software/technological platform, as software components, or services based on software components. The symbiosis of this software can influence the health of a SECO. The influence that the software components have on the SECO health is independent of the actor health. An example would be an actor that creates a software component that enhances the component interoperability and increases the use of the platform and thus contributing to the SECO health. At the same time, this actor might not have a successful revenue model for this software component and end up loosing a big part of the invested effort. The actor will have a negative influence on the SECO health because of low productivity and possibly robustness, while the software component will have a positive influence.

One additional difference of SECOs to BECO/natural ecosystems is that in SECOs there is an entity organizing and managing the ecosystem, the orchestrator. The orchestrator, whether a for-profit organization or an OSS community, is typically managing the ecosystem by running the platform and creating rules and processes for actors and software. The orchestration of the SECO thus has a significant effect on the health of the ecosystem.

The proposed SECO health framework can be seen in Figure 1. We depict three main components that affect the SECO health: (i) the actors, (ii) the software and (iii) the orchestration. In (i), we separate between the individual health of an actor and the health of the network of actors and similarly in (ii) between the individual component health, the ecosystem platform health and the software network health.

4.1 Individual Actor Health

The health of the individual actors influences the overall health of the ecosystem. The actor health can be measured in similar terms to a BECO actor. The actor's productivity and robustness influence the ecosystem. The active participation and engagement of actors brings value to the ecosystem, while the actor's robustness increases the probability that the actor exists and remains involved in the ecosystem activity in the future. If the SECO is a proprietary ecosystem or consists of for-profit organisations, the partner health measures of Hartigh et al. [23] can be directly applied. If in the OSS domain, the actor health can be

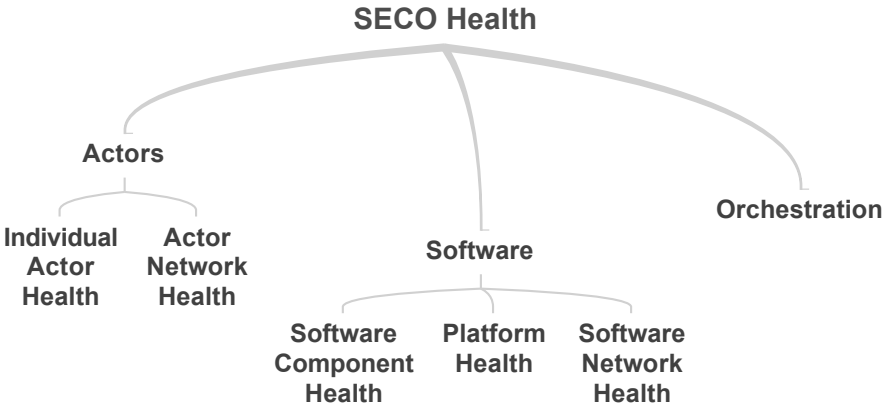


Fig. 1. The SECO health framework breakdown.

assessed in a way similar to Wahyudin et al [29]: measuring the actor activity in the ecosystem (commits, mailing list activity etc.). In that case, an indication of actor robustness is the active participation in the ecosystem over a long period of time. An actor being an active participant in the ecosystem for a long period of time has lower probability of dropping out of the ecosystem than an actor that recently started contributing to the ecosystem.

4.2 Actor Network Health

The network of actors and their interaction plays an important role in the SECO health. The PRN measurements are applicable here, so is the network health perspective of Hartigh et al. [23]. Additionally, the individual actor health may be weighted according to the role of the actor in the network. A keystone with low productivity or robustness will have greater effect in the ecosystem than a niche player with low productivity or robustness.

4.3 Software Component Health

The health of a software component can be measured in terms of, among others, (i) reliability, (ii) availability, (iii) modifiability and prevention of ripple effects, and (iv) interoperability, the ability to interact with, to the extent applicable, the platform and other components. In SECOs, the software components are, in most cases, also the *products* of the ecosystem. The health of such a software component is also influenced by the relative demand and product quality, e.g., how popular is the product and how it is performing in comparison to possible alternatives. This demand is also affected by whether the product is internal, i.e., products intended for use mainly by the ecosystem actors, e.g., the technological platform or external, i.e., products consumed externally to the ecosystem.

4.4 Platform Health

The health characterization of the software components above can be applied to the technological platform of a SECO, since it is a software component itself. However, the technological platform, might have an additional role: depending on how the SECO is organized and managed, the platform reflects possible orchestration actions (rules, processes, or management decisions). The measurement of the platform health should not reflect how the orchestration affects the SECO health (as this is reflected in the orchestration influence on SECO health seen below), but the effectiveness of applying the orchestration actions.

4.5 Software Network Health

The software components are connected and interacting with other components in the ecosystem forming the software network. Graph measures such as connectivity and clustering coefficient show to what extent the components interact [30]. Additionally, the categorization of the activity of hubs into keystone and dominator indicate the level of healthy interaction. Analogous to the Iansiti descriptions in the previous section, an example of keystone activity can be a component that provides interfaces to parts of its functionality for the neighboring components to consume, while in a dominator activity the component would intent to take over functionality of the neighboring components.

4.6 Orchestration Influence to Health

The orchestrator can monitor the health of the ecosystem and take measures to promote ecosystem health if necessary. This requires that the orchestrator has a good overview of the ecosystem and is consulting effective measurements (e.g., ecosystem health). Additionally, the orchestrator can act by creating/refining rules and processes for the actors, communicating plans to the actors (e.g., by road-mapping), organizing the ecosystem development through, e.g., release management, making changes to the platform and other software components, changing the revenue model for internal products, and controlling the actor population and motivation by modifying the model by which the actors participate in the ecosystem. The orchestration of a SECO, i.e., the actions of the orchestrator, possibly based on monitoring and evaluation, influences SECO health.

4.7 Other Influences on SECO Health

Additionally, there might also be influences on the SECO health that are external to the ecosystem. This kind of influences are referred to as "(external) perturbation" in the literature [28, 22, 26, 27] and are disturbances that are outside the control of the ecosystem actors. Influences of this kind might be the establishment or rise of a competitive ecosystem or a radical technological or legal change.

5 Threats to Validity and Future Work

The wider ecosystem health literature used in this study was identified through the references in the SECO health literature as our focus was literature that influenced the SECO health literature. As already mentioned, the literature on each field (apart from SECO) is not necessarily the representative or most influential work in the field. Identification of the most influential work and possible literature mapping of the health in each of the fields (BECO, natural ecosystem, OSS/FLOSS) might bring perspectives into the SECO health that have been overlooked. Additionally, we speculate that the influence of the difference fields to the SECO health is not necessary reflected in the number of papers appearing in this work. Natural ecosystem have had a greater impact on SECO health concepts, but most of it is indirect through the health of BECOs.

Moreover, the proposed conceptual framework, at this point, does not go into detail on the different kinds of actors. An expansion of the model would further analyze on the nature of the actors, e.g., developing companies, resellers, value-adding-resellers, and possibly include their influence on health. Additionally, although the model included products, it did not include customers or end-users. The influence of entities of this kind could be discussed in future work.

6 Conclusion

In this paper, we analyzed the concept of software ecosystem (SECO) health. In order to define SECO health and its measurement, we examined the SECO health literature, a literature body of 13 papers touching upon the concept of SECO health. We identified that the health research is mainly inspired by three fields: business ecosystems (BECO), natural ecosystems, and open source software, with BECO being the main source of inspiration in 11 out of the 13 SECO health papers. We reviewed the wider ecosystem health literature, consisting of 23 papers, explained how they define and measure the health of an ecosystem and concluded with two contributions: (i) We identify two differences between the SECO and business and natural ecosystems: (a) they perceive products in the ecosystem differently. BECOs and natural ecosystems perceive actors as a product per se, while in SECOs an actor produces software components or services. (b) SECOs have an orchestrator entity managing the ecosystem, something that does not appear in the BECO/natural ecosystem literature. (ii) We propose a logical framework for defining and measuring the SECO health consisting of the health of (a) each individual actor, (b) network of actors, (c) each individual software component, (d) platform, (e) software network, and (f) orchestrator. The purpose of this study is to create a discussion on the particularities of SECO health and bring the community closer to a measurable way of defining the health of software ecosystems.

Acknowledgements

This work has been partially funded by the Connect2Care project[4].

References

1. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on. (may 2009) 187 –190
2. Bosch, J., Bosch-Sijtsema, P.: From integration to composition: On the impact of software product lines, global development and ecosystems. Journal of Systems and Software **83**(1) (2010) 67 – 76
3. Messerschmitt, D., Szyperski, C.: Software ecosystem: understanding an indispensable technology and industry. MIT Press Books **1** (2003)
4. Lungu, M., Lanza, M., Gîrba, T., Robbes, R.: The small project observatory: Visualizing software ecosystems. Science of Computer Programming **75**(4) (2010) 264 – 275 Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008).
5. Manikas, K., Hansen, K.M.: Software ecosystems – A systematic literature review. Journal of Systems and Software **86**(5) (2013) 1294 – 1306
6. Mizushima, K., Ikawa, Y.: A structure of co-creation in an open source software ecosystem: A case study of the eclipse community. In: Technology Management in the Energy Smart World (PICMET), 2011 Proceedings of PICMET '11:. (august 2011) 1 –8
7. Boucharas, V., Jansen, S., Brinkkemper, S.: Formalizing software ecosystem modeling. In: Proceedings of the 1st international workshop on Open component ecosystems. IWOCE '09, New York, NY, USA, ACM (2009) 41–50
8. Viljainen, M., Kauppinen, M.: Software ecosystems: A set of management practices for platform integrators in the telecom industry. In Regnell, B., Weerd, I., Troyer, O., Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M.J., Szyperski, C., eds.: Software Business. Volume 80 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2011) 32–43 10.1007/978-3-642-21544-5_4.
9. van den Berk, I., Jansen, S., Luinenburg, L.: Software ecosystems: a software ecosystem strategy assessment model. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ECSA '10, New York, NY, USA, ACM (2010) 127–134
10. Jansen, S., Brinkkemper, S., Finkelstein, A.: Business network management as a survival strategy: A tale of two software ecosystems. In: First International Workshop on Software Ecosystems (IWSECO-2009), Citeseer (2009) 34–48
11. Denscombe, M.: The good research guide. Open University Press (2010)
12. Costanza, R.: Toward an operational definition of ecosystem health. Ecosystem health: New goals for environmental management (1992) 239–256
13. Schaeffer, D.J., Herricks, E.E., Kerster, H.W.: Ecosystem health: I. measuring ecosystem health. Environmental Management **12**(4) (1988) 445–455

[4] `http://www.partnerskabetunik.dk/projekter/connect2care.aspx`

14. van Angeren, J., Blijleven, V., Jansen, S.: Relationship intimacy in software ecosystems: a survey of the dutch software industry. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems. MEDES '11, New York, NY, USA, ACM (2011) 68–75
15. dos Santos, R.P., Werner, C.M.L.: A proposal for software ecosystem engineering. In: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS (2011) 40–51
16. Jansen, S., Brinkkemper, S., Souer, J., Luinenburg, L.: Shades of gray: Opening up a software producing organization with the open software enterprise model. Journal of Systems and Software **85**(7) (2012) 1495 – 1510
17. dos Santos, R.P., Werner, C.: Treating business dimension in software ecosystems. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems. MEDES '11, New York, NY, USA, ACM (2011) 197–201
18. Kilamo, T., Hammouda, I., Mikkonen, T., Aaltonen, T.: From proprietary to open source-"growing an open source ecosystem". Journal of Systems and Software **85**(7) (2012) 1467 – 1478
19. Dhungana, D., Groher, I., Schludermann, E., Biffl, S.: Software ecosystems vs. natural ecosystems: learning from the ingenious mind of nature. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ECSA '10, New York, NY, USA, ACM (2010) 96–102
20. McGregor, J.D.: A method for analyzing software product line ecosystems. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ECSA '10, New York, NY, USA, ACM (2010) 73–80
21. van Ingen, K., van Ommen, J., Jansen, S.: Improving activity in communities of practice through software release management. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems. MEDES '11, New York, NY, USA, ACM (2011) 94–98
22. Iansiti, M., Levien, R.: The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability. Harvard Business Press (2004)
23. den Hartigh, E., Tol, M., Visscher, W.: The health measurement of a business ecosystem. In: Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting. (2006)
24. Iyer, B., Lee, C.H., Venkatraman, N.: Managing in a small world ecosystem: Some lessons from the software sector. California Management Review **48**(3) (2006) 28–47
25. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Business Review **82**(3) (2004) 68–81
26. Iansiti, M., Levien, R.: Keystones and dominators: Framing operating and technology strategy in a business ecosystem. Harvard Business School, Boston (2004)
27. Iansiti, M., Richards, G.L.: The information technology ecosystem: Structure, health, and performance. Antitrust Bull. **51** (2006) 77
28. Rapport, D., Costanza, R., McMichael, A.: Assessing ecosystem health. Trends in Ecology & Evolution **13**(10) (1998) 397–402
29. Wahyudin, D., Mustofa, K., Schatten, A., Biffl, S., Tjoa, A.M.: Monitoring the health status of open source web-engineering projects. International Journal of Web Information Systems **3**(1/2) (2007) 116–139
30. Hansen, K.M., Manikas, K.: Towards a Network Ecology of Software Ecosystems: an Analysis of two OSGi Ecosystems. In: Proceedings of the 25th International Conference on Software Engineering & Knowledge Engineering (SEKE'2013). (2013)

Paper 3

Towards a Network Ecology of Software Ecosystems: an Analysis of two OSGi Ecosystems

Hansen, K. M. and Manikas, K. (2013). Towards a Network Ecology of Software Ecosystems: an Analysis of two OSGi Ecosystems. In *Proceedings of the 25th International Conference on Software Engineering & Knowledge Engineering (SEKE'2013)*, pages 326–331

Towards a Network Ecology of Software Ecosystems: an Analysis of two OSGi Ecosystems

Klaus Marius Hansen and Konstantinos Manikas
University of Copenhagen
Department of Computer Science (DIKU)
Copenhagen, Denmark
{klausmh,kmanikas}@diku.dk

***Abstract*—"Software ecosystems" are gaining importance in commercial software development; the iPhone iOS and Salesforce.com ecosystems are examples of this. In contrast to traditional forms of software reuse, such as common platforms or product lines, software ecosystems have a heterogeneous set of actors sharing and collaborating over one or more technological platforms and business model(s) that serve the actors. However, little research has investigated the properties of actual software ecosystems.**

In this paper, we present an exploratory study of software ecosystems using the formalizations and metrics of the "network ecology" approach to the analysis of natural ecosystems. In doing so, we mine the Maven central Java repository and analyze two OSGi ecosystems: Apache Felix and Eclipse Equinox. In particular, we define the concept of an ecosystem "neighborhood", apply network ecology metrics to these neighborhoods (including a keystone index that identifies the importance of elements in the ecosystem), and compare the ecosystems.

***Index Terms*—software ecosystems; dependency structure; dependency graphs; network ecology**

I. Introduction

Software ecosystems (SECOs) arguably present an effective way of constructing large software systems on top of a software platform by composing components developed by actors internal and external to the organization developing the platform [1]. In this setting, software engineering also takes place outside the traditional borders of software companies to a group of companies, private persons, or other legal entities.

This differs from traditional outsourcing techniques in that the organization developing the platform does not necessarily own the software produced by contributing actors and does not hire the contributing actors. All actors, however, coexist in an interdependent way, an example being the iOS ecosystem in which Apple provides a platform for applications and review of applications in return for a yearly fee and 30% of revenues of application sale[1]. This is a parallel to the natural ecosystems where the different members of the ecosystems (e.g., the plants, animals, or insects) are part of a food chain where the existence of one species depends on the rest.

While examples of successful software ecosystems exist, little empirical research in real software ecosystems exists [2]. In this paper, we study the software ecosystem created around Apache Maven[2], a build automation tool. A characteristic of Maven is that it automatically downloads project dependencies from online repositories. One of the main repositories that Maven consults is the Maven central repository[3], Maven Central, from which we have obtained the repository artifacts' metadata. In particular, we investigate the subset of Maven central that relates to these Apache Felix and Eclipse Equinox OSGi software ecosystems [3].

In the analysis of the ecosystems, we take inspiration from the analysis of natural ecosystems as graphs through "network ecology" [4]. In particular, we investigate software ecosystems using metrics from network ecology. Furthermore, we apply the PageRank [5] algorithm to the structure of software ecosystems.

Our research in this paper is exploratory; the underlying research questions being:

- How can metrics and formalizations of the "network ecology" approach to natural ecosystem analysis provide insight into the structure of software ecosystems?
- How can we analyze and compare different software ecosystems that may be subsets of a wider software ecosystem?

While we do not answer the questions fully, this paper provides initial insight into the answers.

II. Background

A. Apache Maven

Apache Maven [6] is a project management tool, primarily used for Java projects. A central concept in Maven is the "Project Object Model" (POM) that models a project among others allows Maven to build, package, upload to a remote repository, and generate web pages. The main uses of POMs are thus i) to enable building projects and ii) to enable resolution of dependencies in projects. POM files define *artifacts* (i.e., the software components of a project), *versions* of artifacts, and *groups* containing artifacts.

Maven Central[4] is a web service that collects artifacts (e.g., JAR files) and their description in form of POMs. We consider Maven Central as a software ecosystem (referenced to as the Maven ecosystem) with Apache Maven as the common technological platform. Because of the explicitness

[1] http://developer.apple.com/programs/ios/distribute.html
[2] http://maven.apache.org/
[3] http://search.maven.org
[4] http://search.maven.org/

of the dependencies, the Maven ecosystem allows the study of artifacts as a graph. In this paper, we study two ecosystems that are subsets of the Maven ecosystem: Apache Felix and Eclipse Equinox. Both ecosystems, as much as their artifacts, form part of the Maven ecosystem with potential interdependencies. In order to study each of the two software ecosystems, it is necessary to define their size and borders. For this reason we define and calculate the software ecosystem *neighborhood* for each sub-ecosystem.

Given the artifacts in the groups of the OSGi frameworks ("org.apache.felix" and "org.eclipse.equinox"), we follow their dependencies and find artifacts that depend on these artifacts to calculate an ecosystem neighborhood of increasing depth, where for each depth level we add the artifacts depending on artifacts already added or that artifacts already added depend on.

We calculate the neighborhood, A_d = NEIGHBORHOOD(S, d), of a set of artifacts, S, at distance d as follows based on a seed, $S := \{s_1, s_2, ..., s_n\}$, and distance, d:

1) $A_0 := S$
2) $A_{i+1} := A_i \cup \{a \mid \exists b \in A_i : a \text{ depends on } b\} \cup \{b \mid \exists a \in A_i : a \text{ depends on } b\}$

Concretely, we calculated NEIGHBORHOOD(F, i), where F is the set of artifacts in the Apache Felix group (`org.apache.felix`), $i = 0, 1, ...$ and NEIGHBORHOOD(E, i), where E is the set of artifacts in the Eclipse Equinox group (`org.eclipse.equinox`) , $i = 0, 1, ...$.

B. Network ecology

A common theme in ecology of natural ecosystems is a focus on interactions and relationships among living individuals, between predator and prey. "Network ecology" [4], models food webs as *networks* or *graphs* of species and interactions. The networks are often directed to indicate, e.g., energy flow between individuals (e.g., from prey to predator) and are sometimes weighted. Figure 1 shows a simplified example of a food web with one predator (A) and two prey (B and C). Network ecology applies both classical graph/network metrics (e.g., connectance) and novel metrics to food networks (e.g., keystone index); see Section III.

Of particular interest in ecosystems are *keystone species* that have a large effect on their environment compared to the number of members of the species in the environment. In the example, species A may be a keystone since it has effect on both B and C.

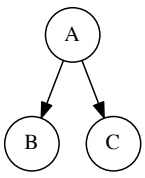


Fig. 1. Simple food web example

In our application of network ecology to the study of software ecosystems, we draw a parallel between species in a natural ecosystem and actors and software artifacts in software ecosystems. Species prey on other species and in this way energy flows from prey to predator. Actors produce software artifacts and software artifacts consume other software artifacts via usage dependencies. And just as the co-existence of species in a natural ecosystem determines the health of the natural ecosystem, the co-existence of actors and software artifacts in a software ecosystem determins the health of the software ecosystem. We thus apply energy flow analysis of natural ecosystems to dependency analysis in software ecosystems. In this our study, we analyzed software ecosystem actors only in terms of their technical products, i.e., the software artifacts, and thus our study is of software dependencies.

In our case, the species relationships from network ecology are usage dependencies among software artifacts and, thus, at runtime, control and data may flow from depender to dependee. Furthermore, our networks are always directed (in the direction of dependencies) and without weight (or equivalently with equal weight).

III. CHARACTERIZING NETWORK ECOLOGIES

In this section, we present the characteristics of network ecologies that we apply in our software ecosystem analysis. We divide the characteristics into *global* characteristics that we calculate for a complete network (Section III-A), and *local* characteristics that we calculate for individual nodes. Furthermore, we apply a page rank algorithm (Section III-C) to the network to compare to network ecology characteristics.

A. Global network characteristics

Traditional graph metrics (e.g., number of nodes and edges) can be applied to networks of ecologies. In particular, we consider connectance, $C = \frac{L}{N \cdot (N-1)}$, where L is the number of relationships in the network, dependencies in our case, and N is the number of nodes in the network, artifacts in our case. Connectance measures the proportion of all potential relationships that are present in the network. Since our network is directed and has no self-relationships, the maximum number of relationships is $N \cdot (N-1)$.

Applying the connectance metric to the Felix and Equinox ecosystems would provide a view on the connected the whole ecosystem is. This, apart from assisting in describing the ecosystem, can measure similarity of the ecosystems as far as connectance goes.

B. Local network characteristics

One of the basic properties of a node is the number of connections this node has to its neighboring nodes, the connection degree of the node. The *connection degree* D of a node i is $D_i = D_{in} + D_{out}$ where D_{in} is the number or nodes depending on node i and D_{out} the number of nodes that node i depends on. The degree of a node is an elementary property that shows how much a node is interacting with its neighboring nodes. Similarly, if we take into consideration the dependency

direction, the *orientation* of a node is $D'_i = D_{in} - D_{out}$. This metric may be used to measure the interaction a node has with its neighboring nodes in directed graphs. If an artifact has a negative orientation, the number of artifacts that it depends is larger than the number of artifacts that depends on it.

The *clustering coefficient* gives further insight into how nodes connect in the graph. The clustering coefficient of a node shows how close this node is to becoming a complete graph with its neighbors. The clustering coefficient of a node i is as follows:

$$CC_i = \frac{L_i}{N_i \cdot (N_i - 1)}$$

where L_i is the number of links between the neighboring nodes N_i of node i. The *global clustering coefficient* for the whole graph is calculated from the clustering coefficient of each node:

$$\bar{C} = \frac{1}{n} \sum_{i=1}^{n} C_i$$

We use the clustering coefficient calculation in our study to analyze the dependencies of the artifacts in each ecosystem. Additionally, we use the calculation to of compare the two ecosystems.

Jordán et al. introduced a reliability-theoretic approach to characterizing keystone indices [7]. They base their keystone index on the concept of a *bottom-up* keystone index and a *top-down* keystone index. The bottom-up keystone index of a species i, $K_b(i)$, is a measure of secondary extinctions because of bottom-up effects if the species i is removed. Here, "bottom-up effects" refer to effects going from prey to predator if the species is removed.

The bottom-up index is defined as:

$$K_b(i) = \sum_{j \in P(i)} \frac{1}{m_i(j)} \cdot (1 + K_b(j))$$

where $P(i)$ are the direct predators of i, $m_i(j)$ the number of direct prey of $P(i)$, and $K_b(j)$ is the bottom-up keystone index for j. The top-down index, $K_t(i)$ is calculated by reversing the species relationships and calculating an index similar to above. Finally, the keystone index is calculated as:

$$K(i) = K_b(i) + K_t(i)$$

Naturally, if species i have no direct predators, their bottom-up keystone index $K_b(i)$ is zero and the keystone index is calculated only by the top-down approach. For the simple example in Figure 1, the keystones indices are $K(A) = 2$ and $K(B) = K(C) = 0.5$.

We adapt the keystone index in the study of the two software ecosystems where instead of a network of preys and predators we have an artifact dependency graph. In this case, the keystone index indicates the importance of an artifact, i.e. what effect the removal of that artifact would have to the software ecosystem.

C. PageRank

The PageRank algorithm [5] calculates the weight of a web page p_i by summing the weights of each page pointing to p_i normalized by the number of links pointing out of that page and adjusted by a dumping factor constant d. PageRank can be explained using the behavior of a hypothetical web surfer who either i) chooses to follow a link at random from a web page with probability d, or, ii) at random chooses any page to go to with probability $1 - d$. The PageRank of a web page p_i, $P(p_i)$ is then the probability of being on that web page that this iterative procedure converges to. The PageRank has the following property:

$$P(p_i) = \frac{(1-d)}{N} + d \cdot \left(\frac{P(p_1)}{C(p_1)} + ... + \frac{P(p_n)}{C(p_n)} \right)$$

where p_1 to p_n are the pages pointing at page p_i, $C(p_i)$ the number of links going out of page p_i and N the total number of pages.

Apart from ranking webpages, PageRank, can be applied to rank nodes in directed graphs. In this study we apply PageRank to estimate weights of the artifacts in the studied ecosystems, with the page links implying artifact dependency. In that sense, an artifact with high PageRank would either have a high number of artifacts or artifacts with high PageRanks depending on it. Such an artifact, will arguably have a large effect on the ecosystem compared with the total number of artifacts. In doing so, we follow an approach close to Bhattacharya et al. [8] who define a "NodeRank" for graph-based representations of software based on PageRank.

Our hypothesis is, that the artifacts with high PageRank would also be "keystone species" in the software ecosystem. For the PageRank ranking of both Apache Felix and Eclipse Equinox software ecosystems, we used a fixed damping factor, d, of 0.85.

IV. Data gathering and processing

We processed the set of 286,922 POM XML files of Maven Central as of 2012-01-24. In the processing, we identified an artifact by group id and artifact id, i.e., we disregarded version numbers of artifacts for reasons of scalability. For each POM file, we output the group id and artifact id and for each dependency declared, we output the relationship between the processed artifact and its dependency.

In total, we extracted 155,127 unique artifacts in 5,676 groups and 495,020 unique dependencies among these artifacts. A number of POM files (36) were empty or contained errors and a number of dependencies (93,455) were not found in the repository.

For the complete set of artifacts and dependencies, we calculated the neighborhoods of the groups org.apache.felix and org.equinox.felix. Table I shows this together with summary statistics of the neighborhoods.

V. Results

In the following, we present the results of applying the metrics of Section III to the neighborhoods of Felix and Equinox. Table I shows the global metrics.

	Felix			Equinox		
Distance	# artifacts	# dependencies	connectance	# artifacts	# dependencies	connectance
0	25	29	0.048333	19	7	0.020468
1	110	138	0.011510	80	69	0.010918
2	7772	8036	0.000133	635	679	0.001687
3	13638	25414	0.000137	9481	10037	0.000112
4	15629	28222	0.000116	14625	25082	0.000117
5	15923	28649	0.000113	15797	26778	0.000107
6	15983	28725	0.000112	15967	26983	0.000106
7	15984	28726	0.000112	15992	27008	0.000106
8	15984	28726	0.000112	15992	27008	0.000106

TABLE I
SUMMARY STATISTICS OF CALCULATED NEIGHBORHOODS

Analyzing the neighborhoods for both ecosystems, we notice that the number of artifacts and dependencies remain the same for distances greater than seven. This implies that the minimum distance of any two indirectly connected artifacts in the Felix or Equinox ecosystems is not more than seven. Therefore, the neighborhood distance of seven is the border of both ecosystems within the whole Maven ecosystem.

A. Network ecology metrics

In the figure 2, we show a log-log histogram for Felix for connection degree $D < 100$, which is the majority of the artifacts (for Felix 15,956 out of 15,984 and Equinox 15,965 out of 15,992). For the Felix degree distribution, the maximum count appears in degree 1 with 7,950 artifacts while the highest degree is 3932 with count of 1 artifact. The artifacts with degrees 1, 2 and 3 account for 80% of the total number of artifacts. For the Equinox group, degree 1 has the maximum count of artifacts, 8,235, while the highest degree is 3,519. Similarly, degrees 1, 2 and 3 account for 82% of the total artifacts. In both groups, the artifact with the highest degree is *junit*. The degree for the artifacts of Felix has an average of 3.59 and a standard deviation of 1,097.74, while Equinox has an average of 3.38 and a standard deviation of 904.63. From the degree distributions, we can verify the similarity in the dependency relations noted for the connectance results of the high distance neighborhoods, with Felix having slightly more dependencies.

Tables II and III display the artifact that have top orientation rankings for Felix and Equinox. In both groups, the top orientation ranking is almost identical (the order for "testing" and "maven-plugin-api" changes) for the top connection degrees. Taking into consideration that the average orientation in both groups is zero, the orientation metrics reveal a tendency in the high ranked connection degree artifacts to have close to zero dependencies to other artifacts, i.e. equally high orientation. This supports the idea of certain artifacts being the center of a cluster or the ecosystem with all the rest of the artifacts depending on them. This enhances the interest of studying the keystone and PageRank metrics.

The clustering coefficient analysis shows that there is a low number of nodes that had a non-zero clustering coefficient (20 out of the 15,984 artifacts for Felix and 4 artifacts from the total 15,992 for Equinox). Out of the 20 non-zero clustering

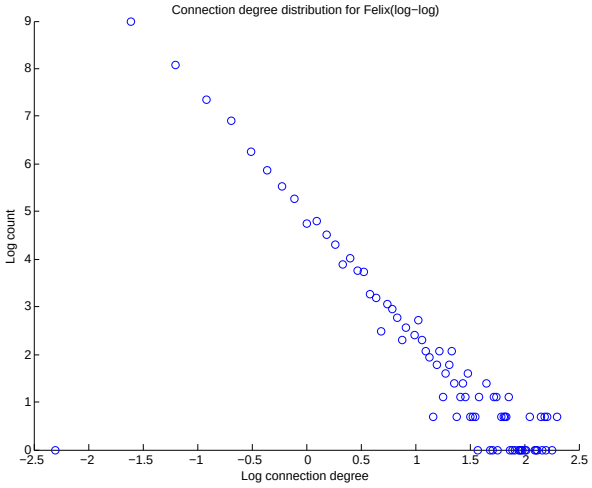


Fig. 2. Connection degree distribution for Felix (log-log;$D < 100$)

org.apache.felix			
Group id	artifact id	degree	orientation
junit	junit	3932	3930
javax.servlet	servlet-api	770	770
org.slf4j	slf4j-api	489	489
org.easymock	easymock	368	368
org.mockito	mockito-all	320	320
commons-io	commons-io	305	305
org.scala-lang	scala-library	235	235
stax	stax-api	234	234
xerces	xercesImpl	202	198
org.apache.maven	maven-plugin-api	196	196
org.testng	testng	198	190

TABLE II
TOP NODES ACCORDING TO THEIR CONNECTION DEGREE AND ORIENTATION FOR ORG.APACHE.FELIX

coefficients for Felix, only 4 artifacts do not belong to the Felix group, while all 4 of the resulting artifacts for Equinox belong to the Equinox group. We may interpret this as Felix having a higher level of reciprocity and thus, according to [9], being a more *open* software ecosystem than Equinox. However, the findings are not adequate to fully support this statement. Table IV shows the top clustering coefficient artifacts for the Felix group and Table V shows the same for the Equinox group . The global clustering coefficient is 0.0002047 for Felix and

org.eclipse.equinox			
Group id	artifact id	degree	orientation
junit	junit	3519	3517
javax.servlet	servlet-api	646	646
log4j	log4j	568	558
org.slf4j	slf4j-api	364	364
org.easymock	easymock	354	354
commons-logging	commons-logging	335	331
org.eclipse.core	runtime	272	268
org.scala-lang	scala-library	263	263
stax	stax-api	262	262
commons-lang	commons-lang	231	231

TABLE III
TOP NODES ACCORDING TO THEIR CONNECTION DEGREE AND ORIENTATION FOR ORG.ECLIPSE.EQUINOX

0.0000377 for Equinox.

The fact that both ecosystems have little clustering is in accordance with the numbers from the previous section where the majority of the artifacts (80% for Felix, 82% for Equinox) had a sum of 3 or less incoming and outgoing dependencies.

org.apache.felix		
Group id	artifact id	CC
org.apache.felix	org.apache.felix.ipojo.handler.white.board.pattern	0.5
org.apache.felix	org.apache.felix.ipojo.handler.extender.pattern	0.5
org.apache.felix	org.apache.felix.ipojo.handler.extender	0.5
org.apache.felix	org.apache.felix.ipojo.arch	0.5
org.apache.felix	org.apache.felix.ipojo.handler.temporal	0.25
org.apache.felix	org.apache.felix.ipojo.composite	0.25
org.apache.felix	org.apache.felix.ipojo.handler.whiteboard	0.167
org.apache.felix	org.apache.felix.scr.generator	0.119
org.apache.felix	org.apache.felix.ipojo.manipulator	0.119
org.apache.felix	org.apache.felix.ipojo.arch.gogo	0.1

TABLE IV
TOP CLUSTERING COEFFICIENTS FOR ORG.APACHE.FELIX

org.eclipse.equinox		
Group id	artifact id	CC
org.eclipse.equinox	preferences	0.5
org.eclipse.equinox	app	0.0833
org.eclipse.equinox	registry	0.01818
org.eclipse.equinox	common	0.00215

TABLE V
TOP CLUSTERING COEFFICIENTS FOR ORG.ECLIPSE.EQUINOX

Table VI and Table VII show the top 10 keystone indices (cf. Section III-A). For both Eclipse and Equinox, JUnit and Hamcrest (upon which JUnit depends) are outliers. For Felix, 3,931 artifacts depend on JUnit (6 on Hamcrest) and for Equinox, 3,518 artifacts depend on JUnit (and 6 on Hamcrest). Further down the list, and of interest, Eclipse-related artifacts appear for both Felix and Equinox, indicating that these are important in both ecosystems.

Scatter plotting of the keystone indices on a log-log scale (not shown), indicates visually that the keystone indices follow a power law distribution.

org.apache.felix		
Group id	artifact id	K
org.hamcrest	hamcrest-core	4285.85
junit	junit	4262.06
javax.servlet	servlet-api	662.97
org.slf4j	slf4j-api	483.07
xml-apis	xml-apis	396.62
org.easymock	easymock	317.17
commons-io	commons-io	288.47
org.eclipse.equinox	common	264.29
org.testng	testng	245.33
org.eclipse.core	runtime	227.88

TABLE VI
TOP KEYSTONE INDICES FOR ORG.APACHE.FELIX

org.eclipse.equinox		
Group id	artifact id	K
org.hamcrest	hamcrest-core	4085.94
junit	junit	4057.05
javax.servlet	servlet-api	616.57
log4j	log4j	445.58
org.slf4j	slf4j-api	430.02
org.easymock	easymock	336.70
org.eclipse.core	runtime	305.49
org.eclipse.equinox	common	273.20
commons-logging	commons-logging	259.43
xml-apis	xml-apis	251.65

TABLE VII
TOP KEYSTONE INDICES FOR ORG.ECLIPSE.EQUINOX

B. PageRank

Table VIII and Table IX show the top 10 PageRanks (cf. Section III-C). The Jaccard index (i.e., the size of the set intersection divided by size of the set union) of the sets of the top 10 artifacts for Eclipse Equinox with respect to keystone index and pagerank is high, 0.67, meaning that the sets are similar. For Apache Felix the Jaccard index is even higher, 0,82.

org.apache.felix		
Group id	Artifact id	$PageRank$
junit	junit	0.0904794430569
org.hamcrest	hamcrest-core	0.0773937500317
javax.servlet	servlet-api	0.0143641067466
org.slf4j	slf4j-api	0.0103344297076
xml-apis	xml-apis	0.00711548141087
org.easymock	easymock	0.00684332751987
commons-io	commons-io	0.00616812512782
org.testng	testng	0.00598292281005
org.mockito	mockito-all	0.00474775877824
org.eclipse.core	runtime	0.00460514855267

TABLE VIII
TOP PAGERANKS FOR ORG.APACHE.FELIX

VI. FUTURE AND RELATED WORK

Researchers have previously analysed networks and graphs in the context of software engineering, in particular on social networks [10]. One problem in this context has been incomplete data, something that we circumvent by obtaining a full copy of the Maven Central POMs.

org.eclipse.equinox		
Group id	Artifact id	$PageRank$
junit	junit	0.0862679466078
org.hamcrest	hamcrest-core	0.0739051502318
javax.servlet	servlet-api	0.0132943226829
log4j	log4j	0.00996111702211
org.slf4j	slf4j-api	0.00894363970498
org.easymock	easymock	0.00735198854011
org.eclipse.core	runtime	0.00676022114354
commons-logging	commons-logging	0.00574973950937
org.testng	testng	0.00536452952305
commons-io	commons-io	0.00486674036983

TABLE IX
TOP PAGERANKS FOR ORG.ECLIPSE.EQUINOX

We recently completed a systematic literature review of research of software ecosystems [2] in which we categorized research output according to the taxonomy of Shaw [11] and found little research on empirical models of software ecosystems. Yu et al. [12] studied symbiosis in software ecosystem as a parallel to symbiosis in natural ecosystems, but no research to our knowledge applied concepts from network ecology to software ecosystems.

Future work includes the analysis of all of the Maven POM data as a network rather than looking at subsets such as Apache Felix and Eclipse Equinox. Similarly to adding additional artifacts in the analysis, an analysis that takes versions and time series into account should also be performed. We made the simplifying assumption of abstracting away version numbers for reasons of scalability of the analysis which may impact the validity of our results. Further, a goal of our research is to find measures of health for software ecosystems, something for which the identification of keystones is only one step towards. In the future, we will also evaluate the use of social network analysis to analyze the artifact dependencies.

Additionally, in this paper, we compared the finding from a keystone index algorithm to the PageRank ranking using a damping factor d of 0.85 for the PageRank algorithm. Different values of the damping factor could enhance the precision of the results.

Another interesting extension is to consider series and dynamics of Maven POM data over time. In particular, Jordán and Scheuring [4] also consider dynamics in their work on network ecology. Extending the network ecology analysis to other ecosystems such as the Eclipse Equinox ecosystem with source code or other open source repositories [13] would also be of interest.

VII. SUMMARY

In this paper, we presented an exploratory study of the Apache Felix and Eclipse Equinox software ecosystems. The Maven central repository stores artifacts belonging to the ecosystems and we extracted descriptions and dependencies based on Project Object Model (POM) files. In total, we analyzed 286,922 POM files, extracting 155,127 artifacts with 495,020 dependencies for further analysis. We defined the neighborhoods for both Felix and Equinox identifying the neighborhood distance of seven as the fix point of both software ecosystems in the Maven ecosystem in terms of number of artifacts included.

For these neighborhoods of Felix and Equinox, we applied concepts from "network ecology" that examines natural ecosystems as networks of prey and predators. We calculated the clustering coefficient for each artifact and the global clustering coefficient for each neighborhoods and concluded that both of the neighborhoods exhibit little clustering. We furthermore defined and calculated a keystone index for the neighborhoods and demonstrated that this index appears to follow a power law distribution. Furthermore, we compared the keystone index to the PageRank algorithm and found that for high-ranked artifacts, there was a large overlap according to a Jaccard index.

We have thus tentatively pointed to the use of the keystone index as a way to identify keystone artifacts in a software ecosystem, artifacts that are of particular importance in the health of the ecosystem. Furthermore, by comparing the Felix and Equinox ecosystems, we demonstrated that it is possible to compare two software ecosystems in terms of size and dependency characteristics.

ACKNOWLEDGMENTS

This work was funded by the Net4Care project. We thank Sonatype for providing access to Maven Central POMs.

REFERENCES

[1] J. Bosch, "From software product lines to software ecosystems," in *Proceedings of the 13th International Software Product Line Conference*, ser. SPLC '09. Pittsburgh, PA, USA: Carnegie Mellon University, 2009, pp. 111–119.

[2] K. Manikas and K. M. Hansen, "Software ecosystems – a systematic literature review," *Journal of Systems and Software*, 2013, in press.

[3] The OSGi Alliance, "OSGi service platform core specification, release 5," http://www.osgi.org/Specifications, June 2012.

[4] F. Jordán and I. Scheuring, "Network ecology: topological constraints on ecosystem dynamics," *Physics of Life Reviews*, vol. 1, no. 3, pp. 139–172, 2004.

[5] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[6] T. O'Brien, M. Moser, J. Casey, B. Fox, J. V. Zyl, E. Redmond, and L. Shatzer, *Maven: The Complete Reference*. Sonatype, 2011. [Online]. Available: http://www.sonatype.com/books/mvnref-book/reference/

[7] F. Jordán, A. Takács-Sánta, and I. Molnár, "A reliability theoretical quest for keystones," *Oikos*, pp. 453–462, 1999.

[8] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos, "Graph-based analysis and prediction for software evolution," in *ICSE 2012*. IEEE Press, Jun. 2012, pp. 419–429. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337273

[9] J. te Molder, B. van Lier, and S. Jansen, "Clopenness of systems: The interwoven nature of ecosystems," in *Third International Workshop on Software Ecosystems (IWSECO-2011)*. CEUR-WS, 2011, pp. 52–64.

[10] R. Nia, C. Bird, P. Devanbu, and V. Filkov, "Validity of network analyses in open source projects," in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. IEEE, 2010, pp. 201–209.

[11] M. Shaw, "Writing good software engineering research papers," *Mini-tutorial for Proc ICSE" 03*, 2003.

[12] L. Yu, S. Ramaswamy, and J. Bush, "Symbiosis and software evolvability," *IT Professional*, vol. 10, no. 4, pp. 56 –62, july-aug. 2008.

[13] G. Robles, J. Gonzalez-Barahona, M. Michlmayr, and J. Amor, "Mining large software compilations over time: another perspective of software evolution," in *Proceedings of the 2006 international workshop on Mining software repositories*. ACM, 2006, pp. 3–9.

Paper 4

Evaluating the Governance Model of Hardware-Dependent Software Ecosystems – A Case Study of the Axis Ecosystem

Wnuk, K., Manikas, K., Runeson, P., Lantz, M., Weijden, O., and Munir, H. (2014a). Evaluating the governance model of hardware-dependent software ecosystems – a case study of the axis ecosystem. In Lassenius, C. and Smolander, K., editors, *Software Business. Towards Continuous Value Delivery*, volume 182 of *Lecture Notes in Business Information Processing*, pages 212–226

Evaluating the Governance Model of Hardware-Dependent Software Ecosystems – A Case Study of the Axis Ecosystem

Krzysztof Wnuk[1], Konstantinos Manikas[2], Per Runeson[1], Matilda Lantz[1], Oskar Weijden[1] and Hussan Munir[1]

[1] Department of Computer Science
Lund University, Sweden
http://serg.cs.lth.se
{Krzysztof.Wnuk,Per.Runeson,Hussan.Munir}@cs.lth.se
{oskar.weijden,matilda.lantz.lth}@gmail.com
[2] Department of Computer Science (DIKU)
University of Copenhagen, Denmark
http://di.ku.dk/
kmanikas@di.ku.dk

Abstract. Ecosystem governance becomes gradually more relevant for a set of companies or actors characterized by symbiotic relations evolved on the top of a technological platform, i.e. a software ecosystem. In this study, we focus on the governance of a hardware-dependent software ecosystem. More specifically, we evaluate the governance model applied by Axis, a network video and surveillance camera producer, that is the platform owner and orchestrator of the Application Development Partner (ADP) software ecosystem. We conduct an exploratory case study collecting data from observations and interviews and apply the governance model for prevention and improvement of the software ecosystem health proposed by Jansen and Cusumano. Our results reveal that although the governance actions do not address the majority of their governance model, the ADP ecosystem is considered a growing ecosystem providing opportunities for its actors. This can be explained by the fact that Axis, as the orchestrator and the platform owner, does not address the productivity and robustness of the ecosystem adequately, but has a network of vendors and resellers to support it and some of the governance activities (e.g. communication) are achieved by non-formal means. The current governance model does not take into consideration.

Key words: software ecosystems, governance model, hardware-dependent ecosystem

1 Introduction

Nowadays, the software development effort is rarely constrained to a single company investing into developers, technology, marketing and sales activities [1, 2]. Forming alliances, participating and benefiting from the capabilities offered by

a software ecosystem, or using open source software, are just a few examples of the development strategies that gain importance in software business. These new forms of collaboration via the "sense of community" [3] come at the expense of decreased control and resulting increase of challenges associated with long term planning. Further, the trade-off between being in control and opening up to ecosystem participants range from technical interface issues to business strategies [4]. Software companies that want to be successful in this context need to learn to open up their platforms and interact with other actors on the ecosystem level, while at the same time ensuring that the strategic goals are fulfilled. These companies need to become orchestrators that mainly determine the growth of their ecosystems [2] and govern them.

Several authors have studied software development governance [3, 5, 6] and proposed different governance techniques, e.g. incremental commitment model [7], decision right automation [8], and transaction cost model [9]. Governance in agile software development was also extensively studied [9, 10, 11, 12, 13, 14]. In the field of software ecosystems, the governance of an ecosystem is argued to have an impact on the overall *health* of the ecosystem [1, 4, 15], i.e. "the extent to which an ecosystem as a whole is durably growing opportunities for its members and those who depend on it" [16]. Jansen and Cusumano [1, 2] have developed a governance model aiming at preserving or improving the health of an ecosystem. The model addresses governance strategies according to the three areas of ecosystem health, inspired by Iansiti and Levien [16]: productivity, robustness and niche creation. To the best of our knowledge, no study has reported the results from evaluating this governance model on a *hardware-dependent* software ecosystem, where hardware plays a dominant role in the value creation process and where the customers purchase hardware devices with software installed on them. Software, in this case, is an enabler for functionality and the main driver for extendability, but without underlying hardware it provides little value to the customers.

In this paper, we assess the governance activities performed by Axis, a network video and surveillance camera producer, the orchestrator and the platform owner of the Application Development Partner (ADP) software ecosystem by investigating the following research question:

> *What governance activities are performed by Axis as a platform orchestrator?*

We conducted an exploratory case study collecting data from a series of observations and interviews and applying the above mentioned model of Jansen and Cusumano to assess the governance of Axis in the ADP ecosystem. Our results show that although Axis meets only part of the model aspects, it is considered from the surrounding actors as a valid ecosystem to participate. Finally, our case study shows that some of the aspects in the model should be expanded to include wider perspectives of governance.

The rest of this paper is structured as follows: Section 2 presents background and related work. Section 3 presents the details about the case company and

Section 4 describes the methodology. The results are presented and discussed in Section 5 and the paper is concluded in Section 6.

2 Background and related work

Developing strategies for effective software ecosystems governance and orchestration was outlined on the agenda for software ecosystems research by Jansen *et al.* [3]. Several authors have studied software development governance. Chulani *et al.* [5] outlined definitions and suggested managing value, developing flexibility and controlling risk and change as the main concerns of software development governance, while Bannerman [6] studied software development governance from meta-management perspective. Several approaches for software development governance were suggested, e.g. based on incremental commitment model [7], using decision rights automation [8], linking long-term business with release planning [9], and using the transaction cost approach [17]. Quite a few articles explore software development governance in agile development [9, 10, 11, 12, 13, 14], yet they do not focus on large-scale hardware-dependent contexts. Only one study explored a context of similar size compared to our case company [11]. From the software ecosystem perspective, Baars and Jansen proposed a framework for software ecosystem governance [15], Jansen *et al.* [4] examined the ecosystem governance from the perspective of the openness of an ecosystem and Jansen and Cusumano [1, 2] build on the top of the two previous studies above to create a governance model for the prevention and improvement of software ecosystem health.

Software ecosystem health is closely related to ecosystem governance: the proper governance decisions can increase the ecosystem health while, ecosystem governance can be evaluated by the effect it has on the health of the ecosystem. Related work contains a number of studies about the health of software ecosystems [18, 19, 20, 21].

3 Case description

Axis is the market leader within network video and surveillance cameras [22]. The company is based in Lund, Sweden, but has offices in 41 countries, partners in more than 179 countries and has 1400 employees [23]. Today Axis' profits are mainly related to sales of camera units, utilizing the two-tier business model with indirect sales. Several different actors such as distributors, system integrators and technology vendors are required to provide complete solutions to end customers. As the amount of software in the video surveillance cameras continues to increase and gains more importance, Axis sees the potential in exploring and developing their *hardware-dependent* software ecosystem.

The Application Development Partner (ADP) is one of the three partner programs at Axis, together with the Application Development Service (ADS) and the Gold Application Development Partner (Gold ADP) programs. The

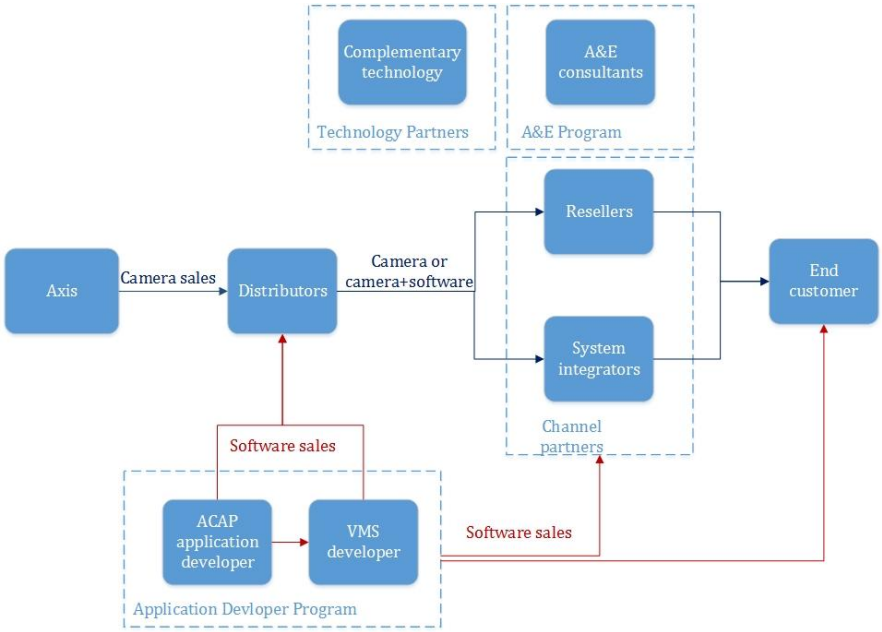


Fig. 1. The software ecosystem surrounding the ACAP, also published in [27]

access to the program is rather easy but in order to advance on to higher levels actively engaged with Axis, companies have to prove that their solutions generate a certain amount of camera sales [22].

The ACAP (Axis Camera Application Platform) ecosystem is based on an open application platform that enables development of third party applications to meet evolving end user needs [24]. Thus, the ecosystem resembles an *application-dependent* ecosystem based on a successful platform i.e. the platform offers customer value without third party applications [25, 2]. Furthermore, the ACAP ecosystem can be considered as screened but free [26]. Axis controls the list of extensions available in the ACAP ecosystem but is not handling any sales, neither offering any joint way of purchasing the ACAP applications. Customers of the ACAP applications are redirected to the websites of the companies developing the ACAP applications in order to download or purchase them. This flow of sales is included in red in Figure 1. Optionally, Axis can offer a licensing system which could also be seen as a part of the extension market. As the main source of revenue for Axis remains camera sales, we consider this ecosystem as *hardware-dependent*.

Axis is the platform leader which has the biggest influence on the decision about the ecosystem, see Figure 1. The main group of external actors constitute the Video Management System (VMS) developers who develop external prod-

ucts, running on servers or similar, and most of them receive image output or control cameras. Both small local and large global system integrators and resellers are among the actors and they could be classified as vendor since they generate profit on selling products produced by the ecosystem. Distributors are also among the actors of this ecosystem but they mostly incorporate software into cameras before selling them [28]. End customers indirectly influence the evolution of the ecosystem via their requirements and needs.

Why Axis? Axis was selected as a case company due to the following reasons: (1) it is a large company that operates globally, (2) it develops embedded systems and provides a case of a *hardware-dependent* software ecosystem, (3) it does not have any direct sales of the products to the end customers, and (4) the end customers do not get directly involved in the development or strategic decisions about the ecosystem and (5) Axis was the market leader also without an ecosystem, which differs from, for example, Android case where Google created the Android ecosystem to enter and become a significant player in the mobile phones market.

4 Research Methodology

As the case company is relatively new in software ecosystems, an exploratory case study method was considered suitable [29]. The main focus of the case study was to understand bridges and barriers in joining the ACAP ecosystem and to investigate the governance model activities. The results regarding the identified motivating and hindering factors are reported in a separate report [27] while this paper focuses on the governance activities.

The study followed the *case study process* proposed by Runeson *et al.* [29]. During the pre-study phase, the company specific literature and related work were studied. Next, ten exploratory interviews among practitioners knowledgeable in the ACAP ecosystem were conducted. The following respondents were interviewed during the pre-study: Global Partner Managers, Product Manager Solutions & Integration Programs, Manager Partner Marketing, Global ADP engineer, Director of System & Services, Senior Engineer for Video Hosting Systems, Business Development Managers, Product Manager API & Components and ADP program manager.

In the next phase, we conducted eight interviews with external developers developing the ACAP applications as well as formal and informal discussions with the Axis employees. Four companies involved in the interviews have an existing ACAP application while the two other companies are not participating in the ACAP ecosystem. Among the participating companies that have ACAP applications, two are quite small with up to 20 employees and two are significantly larger with over 100 employees. These companies offer video analytics solutions based on the ACAP platform. The interviews were transcribed, coded and analyzed by two authors, supervised by more senior authors. Similar statements were put together and abstracted into meta-statements that formed the results statements. The results regarding the ecosystem participation improved

the understanding of the governance activities, including some underlying reasons for performing them. In addition to the above mentioned external partners, 20 practitioners were involved in gathering the data about governance model in both formal meetings and informal discussions. The information gathered during these meetings was systematically stored and analyzed with the similar approach than the interview data. Interesting facts were put together into meta-level facts and compared with the descriptions of the governance activities. The resulting mapping of the performed and not performed activities was presented to the practitioners for validation. By identifying connections and correlations between governance activities, the contextual factors and the identified bridges and barriers to participate, we created an understanding of how governance affect the participation in the ACAP ecosystem.

4.1 Validity analysis

Construct validity refers to possible imperfect operational measures used as a representation of the studied phenomena [29]. There is a risk that the interview questions were not interpreted in the same way by the researchers and the interviewees. To mitigate this threat, we piloted the interview questions on three employees at Axis and two researchers in two iterations. During the interview transcription, potential out of context quotations were discussed and resolved. The list of evaluated governance activities is based on previous work [2] and therefore their suitability as operational measures is confirmed. Finally, the results of the study were presented and discussed with the participants at a workshop.

Internal validity deals with potential confounding factors that may affect studied causal relations [29]. Due to exploratory nature of this study, causal relationships were not considered as the main focus of the study. Therefore, although members of a software ecosystem are often described as closely affecting each other in complex networks [25], the impact of this threat on the validity of the results is minimal.

Reliability refers to the potential biases in the collected data and the analysis methods used by the researchers [29]. We used the governance activity model published earlier, without changing any of the activities. Moreover, we created the interview instrument guided by the existing model and made sure that all relevant aspects were covered in all interviews. However, due to the semi-structured nature of the performed interviews, there are some small differences between the depth of the covered aspects among the interviewees.

External validity discusses the transferability of the findings outside the investigated case. Like for any single case study, threats to external validity remain the main issue in our case. We attempted to mitigate these threats by providing extensive characterization of the studied context [29], including the characterization of the studied ecosystem in order to ease later comparing. Moreover, the studied governance activities are published [1, 2] and by using them we allow other cases to be directly comparable with our results. Finally, we would like to stress the exploratory nature of this study.

5 Governance Activities Performed by Axis

The evaluated governance model for "ecosystem health preservation and improvement" [1, 2] focuses on niche creation, robustness and productivity. The model distinguishes between the software (service) platform and the standard ecosystems, and focuses on the activities that the platform leader should perform in order to improve her position in the software ecosystem. In our case, Axis is the main owner of the software platform which means that the ecosystem is privately owned.

The activities outlined by Jansen *et al.* [1, 2] were compared to Axis' current activities and the results are presented in the subsections that follow. Each activity is marked as **[YES]**, **[NO]** or **[PARTIALLY]** depending on to what degree the activity is performed.

5.1 Activities Connected to Niche Creation

Expand applicability **[YES]** The purpose of the ACAP is to expand the applicability of Axis' cameras to increase sales. Axis is expanding the applicability of the platform by providing access to new features and by releasing more powerful cameras created for new environments. The expansion of applicability should increase the variety of ecosystem participants. This, in turn, may contribute to creating many diverse niches which could allow the ecosystem participants to specialize in their areas, create new products that attract customers to the platform that otherwise would not have been reached [1, 2] and avoid head-on competition [30]. However, as the participants are active within the same industry and provide similar types of applications, the expansion possibilities are limited, causing entry barriers for one of the two studied companies that currently do not develop ACAP applications.

Make strategy explicit **[NO]** None of the interviewees received explicit information about the ACAP strategy and only some respondents stated that they *implicitly* received this information during discussions and collaboration with employees at Axis. Axis has no explicit strategy for ACAP but has transparent relationships with developers. The possible interpretation could be that transparent relationships are enough to ensure niche players about their future position within the ecosystem [1] and create trust among participants towards the platform leader's intention and commitment. This approach seems to be efficient for relatively small number of ecosystems players just like in our context.

We have not identified any trust issues among the ACAP developers participating in the study. One possible explanation could be that all these developers had, prior to joining the ecosystem, a relationship with Axis and described it as good and transparent, indicating increased trust. Also, several companies received the information about Axis' strategy implicitly through contact with Axis personnel. Therefore, it seems that a healthy relationship and transparent communication decreases the need for an explicitly communicated strategy.

Create API **[YES]** Axis has created a collection of API:s connected to the ACAP that reduced compatibility issues, increased the degree of control [1] and

increased the productivity of niche players [19]. Therefore, creating an API was described as one of the benefits and reasons to develop toward the ACAP [27]. The need for an API was fostered by: (1) base technology: several product lines, (2) actors: fragmented customers, and (3) competitors: not offering an internal standard similar to the ACAP.

Co-development **[NO]** Axis does not perform any co-development, i.e. joint development projects with other companies. The lack of co-development has not had any identified effects on this ecosystem. This result could suggest that co-development does not attract niche players in this kind of software ecosystem, which contradicts with the previous studies [1, 2]. Another possible interpretation could be that niche players have knowledge about both the domain and the platform and thus do not need co-development. This contradicts with the viewpoint of Hanssen [31]. Finally, the need for obtaining synergies that can drive innovation, reduce costs and development time [32] may not be that strong in our context.

Develop complementary platforms **[NO]** Axis has no plans to develop complementary platform, thus we consider this activity as not being performed.

Develop new business models **[NO]** Axis focuses on camera sales and utilizes the two-tier sales model. Axis has no requirements regarding the ACAP application sales and distribution. They provide a free licensing system to the users of the platform but at the same time is not involved in sales and distribution of the ACAP applications. Axis offers a licensing for free business model connected to the platform and is not facilitating any other business models. The possible interpretation could be that licensing based business models are a good fit for the environment of this ecosystem.

Axis is restricting third party developers from being a part of their chain of distribution. This has a negative effect on enabling new niches and business opportunities by introducing new business models to third parties, e.g. by introducing a marketplace which enables third party developers to reach customers they would not have reached on their own [1, 2]. Related work by Hagel *et al.* [30] suggested that the platform leader's responsibility is to provide focus through identified business opportunities and forces connected to the ecosystem.

5.2 Activities Connected with Robustness

Create partnership model **[YES]** The ADP (Application Developer Partner) program is an established partnership program offered to all companies interested in developing software for Axis cameras and allows to set up rules for partners in the ecosystem [1, 4]. However, the program is explicitly focused on promoting developers of high volume and broad applications, rather than niche applications, which most ACAP applications are. Thus, the availability of the program is not considered as an incentive for the potential ACAP developers [27].

The requirements to reach the highest partner level are steep, hindering the ACAP developers from advancing to this level due to their size and niched applications. As a result, the support needed to explain the ACAP developers' businesses is blocked (also due to lack of sales) by the inability to advance in the

ADP program. Furthermore, Axis' partner program does not allow independent developers, decreasing the variety of the ecosystem.

Do marketing **[YES]** Axis' main marketing activities are conducted in order to increase cameras sales. Marketing activities towards potential ACAP developers are sporadic and small compared to the marketing of cameras. As a result, the awareness among customers and developers about the ecosystem is not fully explored and may negatively impact the ecosystem participation [1, 2].

The presence of end customer's demand to develop ACAP applications suggests that the customers are aware of the ACAP platform. Moreover, as the majority of the ACAP developers already had a relationship with Axis before developing the ACAP applications [27], the developers' awareness and marketing activities may have only limited effect on participation.

Grow profits **[NO]** Axis is focusing on camera sales and is not interested in increasing the profits by providing ACAP applications. However, one of the requirements to join the gold application partner level is to prove that the applications generate a certain amount of camera sales. Thus, the potential additional revenue streams for ACAP applications are considered insignificant.

Partner development programs **[YES]** These programs could help Axis to strengthen the potentially less productive weak actors that could decrease the health and stability of the ecosystem. Axis' learning center provides training, seminars, classroom training, tools and quick reference help [33] and is accessible for members in the ADP program.

The learning center is not designed as a program, but rather as a source of information, support and training. Axis does not offer any financial support to partners, but the main reason for a development program is to help strengthen members of the ecosystem and that is fulfilled today. The technical expertise delivered by Axis was found to ease the transition to the platform and to improve the perceived quality of communication with developers, which was also considered as one of the reasons to join this software ecosystem [27].

Form alliances **[PARTIALLY]** Axis has existing alliances with many relevant companies within the industry through their partner programs, see Section 3, but the focus of these relationships is not on the ACAP or its applications. Therefore, the opportunities of forming sub-groups of participants or strategic incumbents in a market and in this way increasing the robustness of the ecosystem [1, 16] are not fully explored. The existing alliances within surveillance industry could be utilized for strengthening the ACAP and its ecosystem.

Stabilize API:s **[YES]** Axis has stable API:s that remained unchanged after integration of new features caused by the ACAP introduction. In this regard, Axis complies with the advices published in related work to ensure backward compatibility, simplify software configuration [34] and create consistency which leads to increased trust in the platform [1, 2]. Axis is aware that the API:s are not optimal, but sees it as a higher priority to keep them stable rather than to change them. This strategy pays off as stable API:s were considered as one of the benefits and reasons to join the ecosystem [27].

Raise entry barriers **[NO]** Entry barriers help to ensure that the right companies join the ecosystem and can be used as a mechanism to steer its growth [1]. If entry barriers are too low, the stability of the ecosystem might decrease because of uncontrolled growth and loss of quality (in developers or the components they develop) and thereby the increases risk of an unhealthy ecosystem [19]. Therefore, high entry barriers are a recommended way to increase the quality of an ecosystem [1, 2] by fees, certification programs for the applications and more rigorous screening of customers [35]. However if the barriers become too high they might exclude too many developers and hinder innovation [19].

Axis does not impose high entry barriers to join their application development program: members only have to be a registered company. However, this blocks access for independent developers, for example students. The company does not take any fees or commissions associated with published applications. However, our results suggest that the barriers could be considered as high (not deliberately set by Axis) because of the following reasons: the dependence of external software and other actors, the fragmented customer base of Axis end customers, and the lack of an accessible way to reach the market.

The domain dependence together with the relatively low number of third party developers in the studied ecosystem imply that Axis should facilitate participation and lower entry barriers for newcomers in opposite to what is suggested by Jansen *et al.* [1, 2]. This confirms previous research which indicated that high entry barriers might exclude too many developers [19].

Make partners explicit **[YES]** Axis publishes a list of ACAP developing companies on their company website and thus making the partners explicit [36].

Propagate operation knowledge **[NO]** Axis does not have a systematic way to collect end user experiences, knowledge of in-the-field-performance or feedback [37] related to ACAP and is hence not able to communicate these to other members of the ecosystem. Therefore, we assessed this activity as not being performed. No negative effects of not propagating operational knowledge were found. One possible explanation may be that Axis' two-tier business model reduces direct contact with end customers and the ability to collect such data. Therefore, this task might not be suitable for the platform leader in this ecosystem and may not lead to significant performance improvements [37].

5.3 Activities Supporting Productivity

Organize developer days **[NO]** Before launching ACAP, Axis has hosted a training session for developers in Lund. However, the current arrangements of trainings at Axis do not include the ACAP developers, unless they offer an additional product and hence are qualified. Therefore, the potential benefits, e.g. increased interaction [19], a higher degree of connectedness [19], robustness [19, 16], more internal connections, raised awareness of the platform [1] and increased probability of survival [16] are not fully explored. We discovered that this activity directly effects the participation in this ecosystem [27]. Enabling new players to easily connect and creating external standards to increase compatibility could in this case be also helpful.

Collaborative marketing **[PARTIALLY]** Axis does not systematically perform collaborative marketing efforts [38] with third party developers. On a case by case basis, some forms of collaborative marketing are performed at exhibitions and fairs. Thus, the potential benefits derived from fusions of the products or resource pooling are not fully explored [38].

Create sales partner program and create new sales channels **[NO]** Axis has a channel partner program including companies distributing and selling network video products and solutions. This does not apply to distribution of software or more specifically ACAP applications. Axis does not have any outspoken strategy for how ACAP applications should appear on the market. Thus, the possible increase of sales margins of ACAP software could not be evaluated. One of the possible reasons is that many ACAP developers are relatively small players in the surveillance industry and thus less interesting for Axis. It seems like the opportunity of creating more value by connecting niche players to customers and enabling more revenue for the ecosystem participants [1, 2] (both niche players and the platform leaders) is not fully explored in our case [19].

However, Axis has historically seized opportunities to cooperate with existing customers and provided information and sales support, although, this was done sporadically and through personal connections. As a result, new developers without industry experience or a relationship with Axis would find it difficult to identify which relationships are needed to access the end customers [39]. Creating more established relationship with Axis could reduce the perceived risk [39] and open access to important information and support.

5.4 Remarks from the evaluation

Some interesting and important remarks can be made after our evaluation of the governance model proposed by Jansen and Cusumano [1, 2]. Several activities were confirmed as important and necessary, among them the needs to: expand applicability beyond the current domain, create and keep stable API:s, form partnerships, create partner development programs focused on niche players, support developers by organizing developer days, do marketing and extent current business models with niche players in mind.

At the same time, only 66% of the niche creation activities, 44% of the robustness activities and 25% of the productivity activities are fully performed by Axis. Regarding making the strategy explicit, our results suggest that healthy relationship and transparent communication could be a good surrogate for explicit strategy for a relatively small number of ecosystem players. The lack of co-development and complementary platforms have not had any identified effects on this ecosystem. This result could suggest that: (1) co-development does not attract niche players in this kind of software ecosystem or (2) niche players have knowledge about both the domain and the platform and thus do not need co-development, which contradicts with the viewpoint of Hanssen *et al.* [31]. The lack of new business model development suggests that licensing based business models are suitable for this ecosystem. Due to focus on camera sales and rela-

tively low potential of the ACAP applications revenue stream, Axis seems not to be interested in growing profits from the ACAP ecosystem.

Our results confirm that keeping high entry barriers helps to ensure the quality of the ecosystem but also limits the participation of independent developers and students not employed by companies involved in an ecosystem. Similarly, although Axis does not propagate knowledge about the ACAP ecosystem, we did not found this having any negative effects. This might be either because of the specific nature of the ecosystem or because there were other unofficial channels for propagating knowledge. Finally, the possibilities of creating more value and revenue via partner programs by connecting niche players to customers [1, 2] are not explored by Axis.

To summarize, out of 19 activities in three areas Axis fully performs 8 activities, these are marked as "Yes" and two partly, these are marked as "Partially". Nine activities, marked as "No" in all three areas are not performed. This could be an early indication of signs of low health in the ecosystem. However, the ecosystem is slowly growing in actor size and potential and increasing the value for the connected actors. According to the governance framework, the ecosystem has low or no governance activities supporting productivity, with only one activity partially supported. However the Axis ecosystem is differentiated from most of the ecosystems studied in related work [1, 2] by the fact that the platform orchestrator (i.e. Axis) was the market leader before the ecosystem was created and is not the one supporting the business and revenue models for the actors. Cameras with or without developed software are packed and distributed by a set of distributors, resellers and system integrators, that are external to Axis. Therefore although Axis, as platform owner and orchestrator, does not undertake governance activities to ensure productivity, this task is covered by the network of distributors, resellers and system integrators. An expansion of the model, thus, would be to include activities of vendors and resellers into the productivity section, support unofficial or non-formal channels for knowledge dissemination and explore the role of licensing business models in ecosystems governance. Finally, a necessary addition to the current model could be to consider some activities as *satisfy explained* which legitimates their absence due to specific company or business context conditions.

6 Conclusions

In this study, we focus on the governance of a *hardware-dependent* software ecosystem. More specifically, we evaluate the governance model applied by Axis, a network video and surveillance camera producer that is the platform owner and orchestrator of the Application Development Partner (ADP) software ecosystem. We conducted an exploratory case study collecting data from observations and interviews and applied the governance model for the prevention and improvement of the software ecosystem health proposed by Jansen and Cusumano [1, 2].

Only 66% of niche creation activities, 44% of robustness activities and 25% of productivity activities are fully performed by Axis. Our results reveal that al-

though the governance actions do not address the majority of the applied framework, the ADP ecosystem is considered a growing ecosystem providing opportunities for its actors. This is explained by the fact that Axis, as the orchestrator and the platform owner, does not address productivity and robustness of the ecosystem, but has a network of vendors and resellers to support it and several of the governance activities (e.g. communication) are achieved by non-formal means. The current governance model does not take this into consideration.

In future work, we plan to investigate another *hardware-dependent* software ecosystem to enable meta-analysis and comparison. Moreover, we plan to investigate the impact of the business model utilized by Axis on the governance activities and further explore how Axis can integrate the potential additional revenue stream into this business model.

Acknowledgements: This work is founded by the SYNERGIES project, Swedish National Science Foundation, grant 621-2012-5354. We thank Axis and their partners for their openness during the study.

References

1. Jansen, S., Cusumano, M., Brinkkemper, S.: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar Publishing, Incorporated (2013)
2. Jansen, S., Cusumano, M.: Defining software ecosystems: A survey of software platforms and business network governance. In: The 4th International Workshop on Software Ecosystems. (2012)
3. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on. (2009) 187–190
4. Jansen, S., Brinkkemper, S., Souer, J., Luinenburg, L.: Shades of gray: Opening up a software producing organization with the open software enterprise model. Journal of Systems and Software **85**(7) (2012) 1495 – 1510
5. Chulani, S., Williams, C., Yaeli, A.: Software development governance and its concerns. In: Proc of the 1st international workshop on Software development governance. SDG '08, New York, NY, USA, ACM (2008) 3–6
6. Bannerman, P.L.: Software development governance: A meta-management perspective. In: Proc of the 2009 ICSE Workshop on Software Development Governance. SDG '09, Washington, DC, USA, IEEE Computer Society (2009) 3–8
7. Boehm, B.: A process framework for system and software development governance. In: Proc of the 1st international workshop on Software development governance. SDG '08, New York, NY, USA, ACM (2008) 1–1
8. Kofman, A., Yaeli, A., Klinger, T., Tarr, P.: Roles, rights, and responsibilities: Better governance through decision rights automation. In: Proc of the 2009 ICSE Workshop on Software Development Governance. SDG '09, Washington, DC, USA, IEEE Computer Society (2009) 9–14
9. Vähäniitty, J., Rautiainen, K.T.: Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development. In: Proc of the 1st international workshop on Software development governance. SDG '08, New York, NY, USA, ACM (2008) 25–28

10. Raatikainen, M., Rautiainen, K., Myllärniemi, V., Männistö, T.: Integrating product family modeling with development management in agile methods. In: Proc of the 1st international workshop on Software development governance. SDG '08, New York, NY, USA, ACM (2008) 17–20
11. Lehto, I., Rautiainen, K.: Software development governance challenges of a middle-sized company in agile transition. In: Proc of the 2009 ICSE Workshop on Software Development Governance. SDG '09, Washington, DC, USA, IEEE Computer Society (2009) 36–39
12. Cheng, T.H., Jansen, S., Remmers, M.: Controlling and monitoring agile software development in three dutch product software companies. In: Proc of the 2009 ICSE Workshop on Software Development Governance. SDG '09, Washington, DC, USA, IEEE Computer Society (2009) 29–35
13. Ambler, S.W.: Scaling agile software development through lean governance. In: Proc of the 2009 ICSE Workshop on Software Development Governance. SDG '09, Washington, DC, USA, IEEE Computer Society (2009) 1–2
14. Qumer, A.: Defining an integrated agile governance for large agile software development environments. In: Proc of the 8th international conference on Agile processes in software engineering and extreme programming. XP'07, Berlin, Heidelberg, Springer-Verlag (2007) 157–160
15. Baars, A., Jansen, S.: A framework for software ecosystem governance. In Cusumano, M.A., Iyer, B., Venkatraman, N., eds.: Software Business. Volume 114 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2012) 168–180
16. Iansiti, M., Levien, R.: The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability. Harvard Business School Publishing India Pvt. Limited (2004)
17. Erbas, C., Erbas, B.: Software development under bounded rationality and opportunism. In: ICSE Workshop on Software Development Governance. (2009) 15–20
18. Manikas, K., Hansen, K.M.: Reviewing the health of software ecosystems – a conceptual framework. In: 5th International Workshop on Software Ecosystems (IWSECO). (2013) 33–44
19. van den Berk, I., Jansen, S., Luinenburg, L.: Software ecosystems: a software ecosystem strategy assessment model. In: Proc of the Fourth European Conference on Software Architecture: Companion Volume. ECSA '10, New York, NY, USA, ACM (2010) 127–134
20. Jansen, S., Brinkkemper, S., Finkelstein, A.: Business network management as a survival strategy: A tale of two software ecosystems. In: Proc of the First Workshop on Software Ecosystems 2009. IWSECO '09 (2009) 34–48
21. van Angeren, J., Blijleven, V., Jansen, S.: Relationship intimacy in software ecosystems: a survey of the dutch software industry. In: Proc of the International Conference on Management of Emergent Digital EcoSystems. MEDES '11, New York, NY, USA, ACM (2011) 68–75
22. Axis Communications AB: About axis communications. `http://www.axis.com/corporate/about/index.htm` (last visited, April 2014)
23. Axis Communications AB: Annual report 2013. `http://www.axis.com/files/annual\_reports/2012annual\_eng.pdf` (last visited, April 2014)
24. Axis Communications AB: Participation in ACAP. `http://www.axis.com/corporate/press/industry\_news/article.php?article=090921\_applicationplatform.htm` (last visited, April 2014)

25. Bosch, J.: From software product lines to software ecosystems. In: Proc of the 13th International Software Product Line Conference. SPLC '09, Pittsburgh, PA, USA, Carnegie Mellon University (2009) 111–119
26. Axis Communications AB: Applications ready to meet your needs. `http://www.axis.com/products/video/compatible\_applications/index.php` (last visited, April 2014)
27. Wnuk, K., Runeson, P., Lantz, M., Weijden, O.: Bridges and barriers to hardware-centric software ecosystem participation a case study. Technical report, Lund University, Department of Computer Science, http://serg.cs.lth.se/index.php?id=89149 (2014)
28. Manikas, K., Hansen, K.M.: Software ecosystems – a systematic literature review. Journal of Systems and Software **86**(5) (2013) 1294–1306
29. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering – Guidelines and Examples. Wiley (2012)
30. Hagel, J., Brown, J.S., Davison, L.: Shaping strategy in a world of constant disruption. Harvard Business Review (10) (2008)
31. Hanssen, G.K.: A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. Journal of System and Software **85**(7) (2012) 1455–1466
32. Chesbrough, H.: Open Innovation: The new imperative for creating and profiting from technology. Boston: Harvard Business School Press (2003)
33. Axis Communications AB: Axis' learning center. `http://www.axis.com/academy/` (last visited, April 2014)
34. Viljainen, M., Kauppinen, M.: Software ecosystems: A set of management practices for platform integrators in the telecom industry. In: Software Business. Volume 80 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2011) 32–43
35. Rao, A.R., Ruekert, R.W.: Brand alliances as signals of product quality. MIT Sloan Management Review (1994)
36. Axis Communications AB: The list of the compatible applications. `http://www.axis.com/products/video/compatible_applications/index.php` (last visited, April 2014)
37. van der Schuur, H., Jansen, S., Brinkkemper, S.: The power of propagation: on the role of software operation knowledge within software ecosystems. In Grosky, W.I., Badr, Y., Chbeir, R., eds.: MEDES, ACM (2011) 76–84
38. Bucklin, L.P., Sengupta, S.: Organizing successful co-marketing alliances. Journal of Marketing **57**(2) (1993) pp. 32–46
39. Das, T.K., Teng, B.S.: Trust, control, and risk in strategic alliances: An integrated framework. Organization Studies **22**(2) (2001) 251–283

Paper 5

Characterizing the Danish Telemedicine Ecosystem: Making Sense of Actor Relationships

Manikas, K. and Hansen, K. M. (2013a). Characterizing the danish telemedicine ecosystem: Making sense of actor relationships. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems*, MEDES '13, pages 211–218

Characterizing the Danish Telemedicine Ecosystem: Making Sense of Actor Relationships

Konstantinos Manikas
Department of Computer Science (DIKU)
University of Copenhagen
Njalsgade 128
2300 Copenhagen S
Denmark
kmanikas@diku.dk

Klaus Marius Hansen
Department of Computer Science (DIKU)
University of Copenhagen
Njalsgade 128
2300 Copenhagen S
Denmark
klausmh@diku.dk

ABSTRACT

The use of telemedicine is arguably beneficial even in densely populated areas in reducing cost and increasing efficiency of healthcare. However, the implementation of telemedicine solutions in the healthcare system of Denmark has been perceived as being faced with implementation and interoperability issues, silo solutions, and lack of guidelines and standards. In this paper, we characterise the ecosystem evolved around the telemedicine services in Denmark and study the actors involved in this ecosystem. We establish a method for this study, where we define two actor roles and ways of characterizing actor contributions, and apply the method to the largest healthcare region of Denmark. Our findings reveal an ecosystem that is relatively closed to new actors, where the actors tend to be related to single telemedicine applications, the applications have low connectivity, and the most influential actors of the ecosystem can be characterised as both being beneficial and inhibitory to the ecosystem prosperity.

Categories and Subject Descriptors

K.6.4 [**Management of Computing and Information Systems**]: Software Management—*Software development*; E.1.3 [**Data**]: Data Structures—*Graphs and networks*

General Terms

Standardisation, Management, Measurement

Keywords

Telemedicine ecosystem, software ecosystem, ecosystem actor analysis

1. INTRODUCTION

Telemedicine has been successfully applied in sparsely populated countries [1] and is now increasingly used in more densely populated countries such as Denmark. Telemedicine arguably reduces cost and increases efficiency in the treatment of elderly and patients with chronic diseases [2] and has been shown to reduce mortality and emergency admission rates [3]. However, implementation of telemedical technologies is expensive and hard [4]. Proprietary solutions that do not integrate with existing healthcare systems, standards that are hard to use, application-specific data models, and application silos make interoperability and communication between telemedicine and other healthcare applications hard or non-existent. These are some of the reasons that result in the existing, implemented telemedicine applications today being mainly restricted to small geographical areas, and, although they might be collecting data of wider medical interest, they are often unable to share this data with other systems or utilize data from other systems. Additionally, most of the reasons above contribute to a significant integration costs of new telemedicine applications. This results in that many activities in the telemedicine software industry tend to become the domain of companies with possibilities of higher and long-term investment.

The above underlines the importance of making changes to the existing (organizational and technical) structure of the Danish telemedicine services for telemedicine to be widely and successfully adapted and implemented in the healthcare system. In order to identify points of improvement and restructure, the existing telemedicine ecosystem needs to be studied and its structure in combination with weak points and bottlenecks identified.

In this paper, we analyze the existing organizational and technical structure of the telemedicine ecosystem in Denmark. More specifically, this paper provides the background of the telemedicine ecosystem of Denmark (Section 2), reports on related work and identifies that no previous work has studied the actor relationships in a proprietary software ecosystem (SECO) (Section 3), proposes that the study of the actors of an ecosystem provides information about the prosperity and well-functioning of the ecosystem and provides a method for conducting a study like this in the Danish telemedicine ecosystem (Section 4), analyses the findings from the study of the largest healthcare region in Denmark (Section 5), discusses future work and threats to validity (Section 6), and concludes (Section 7).

2. BACKGROUND

In this section, we provide background on the (Danish)

MEDES'13 October 28-31, 2013, Neumünster Abbey, Luxembourg

telemedicine ecosystem. We first explain the concept of a software ecosystem. Next, we provide the background of the ecosystem under study by analysing the management, administration and financing of the general healthcare services in Denmark and finally define the Danish telemedicine ecosystem and identify particularities in comparison to the general software ecosystem definition.

2.1 Software ecosystems

A software ecosystem (SECO) can be defined as:

> "the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services. Each actor is motivated by a set of interests or business models and connected to the rest of the actors and the ecosystem as a whole with symbiotic relationships, while the technological platform is structured in a way that allows the involvement and contribution of the different actors" [5].

The study of SECOs is inspired by natural ecosystems where different species co-exist and their survival depends on the survival of the rest of the species, effectively forming a food network. An example of a SECO is Apple's iOS where Apple provides interfaces for components of the technological platform that external developers (actors) can use to develop apps and generate a revenue by selling them in the AppStore (thus satisfying a business model). Looking at the definition, and as we noted in the example, we identify three main components in a SECO: (i) a set of actors, (ii) a technological platform, and (iii) a set of business models that serves these actors.

The actors are characterised by their roles in the ecosystem [5]: The *orchestrator* is the organiser of the ecosystem that, in most cases, manages the platform and sets rules and processes. A *niche player* provides value to the ecosystem and, many times, takes part on the management of the ecosystem. *External developers* take advantage of the possibilities the ecosystem provides in developing software solutions and by that indirectly return value to the ecosystem.

2.2 The Danish healthcare system

The Danish healthcare system is divided into (i) the primary sector, that is responsible for preventive healthcare and practicing diagnosis and treatment, and (ii) the secondary and tertiary sectors that provide specialised care. The primary sector consists of private practitioners (e.g., general practitioners (GPs), specialists, and dentists) and municipal/local health service providers, while the secondary and tertiary sectors are mostly concentrated in hospitals. [6].

To manage the healthcare system there are three administrative levels [7]:

- *Local.* This consists of 98 municipalities that are responsible for prevention, health promotion, and rehabilitation.
- *Regional.* Consisting of five regions that are responsible for hospitals, psychiatry, and finance private practitioners (GPs and specialists).
- *National.* The Ministry of Health governs the local and regional provision of healthcare.

When considering the financing, healthcare is part of the Danish welfare state that serves as an example of the "universal model" [8]. All citizens have equal access to a number of services provided by the state while these services are funded by collected tax. The same applies for healthcare: Danish healthcare services are provided to the citizens without direct payment for the service they receive, since healthcare is funded by tax. More specifically there are two different kinds of taxes related to healthcare: the municipality tax paid to the municipalities (that is among others funding the municipalities' healthcare activities) and the "health contribution" tax paid to the state. The regions, that are the main healthcare provider is funded by 80% from the state and 20% from the municipalities [9].

2.3 The telemedicine ecosystem

The main focus of this work is the ecosystem evolved around the telemedicine services of the Danish healthcare. We use the definition of the World Health Organisation to define telemedicine as the "delivery of health care services, where distance is a critical factor, by all health care professionals using information and communication technologies" [10]. Currently there is a number of telemedicine applications used in the Danish healthcare ranging from simple, but effective, applications such as the "Telesår" application for home nurses that makes use of a smartphone to take pictures of patients' feet for remote monitoring of ulcers and send them to a specialist for evaluation, to more complicated systems such as the integrated "KIH" system for remote monitoring of patients with chronic obstructive pulmonary disease (COPD) that includes a device with sensors at the patient's site to collect data to repositories and analyze them for clinicians' view and evaluation.

If we examine the telemedicine ecosystem, we see an ecosystem that provides software solutions aimed at facilitating prevention, diagnosis, and treatment for patients. This is achieved by creating solutions focusing on the patients (or citizens in the case of prevention), clinicians, or both. When we analyze the telemedicine ecosystem according to the SECO components mentioned in Section 2.1, we make the following remarks.

In the set of actors, at a first glance, we identify the healthcare management actors mentioned in the paragraph explaining the Danish healthcare (Section 2.2) as actors of the ecosystem: the Ministry of Health, the National Board of Health, the five regions, and the municipalities. The regions, in collaboration with the National Board of Health and the Ministry of Health, have the role of orchestrator in the ecosystem. Their responsibility is to assure that development within the ecosystem is providing value to the ecosystem (and eventually to the patients/citizens or clinicians). The municipalities might also be involved in the orchestration, but to a smaller extent, as they have fewer healthcare responsibilities. Their main role is to support the regions financially. However, the municipalities, apart from participating in the orchestration, can have a more active participation in the ecosystem by being involved in the projects, e.g., having telemedicine applications implemented in their premises. In this paper, we mainly focus at the role of the regions.

For the needs of characterizing the actor activity of the telemedicine ecosystem we define two specialised actor roles:

- A *host* is typically facilitating the implementation of

telemedicine applications: allows the implementation of telemedicine applications in its organizational premises and serves as a link between the users (i.e., clinicians, patients, citizens). An example of a host can be a hospital having a telemedicine system implemented.

- A *developing actor* is involved in developing a telemedicine application or system. This could, e.g, be a software development company, an infrastructure supplier or a project management/administration consultancy.

The business models that serve the actors in this ecosystem is somewhat different than the Apple iOS example. In contrast to a company developing an application, in the telemedicine ecosystem, the most common way of developing an application or system is to have a project initiated that addresses a specific issue. In such a project, the orchestrator is mostly represented by the region(s) or the state, the users (or part of them) by one or several hosts and the project is developed by one or several developing actors. The activities and actors are funded by the involved regions while the end users (patients, clinicians) are external to the funding process. Following the financial model of the healthcare system (as explained in Section 2.2), the telemedicine ecosystem actors demonstrate a number of symbiotic relationships as they have a specific amount of funding to share within a project and among the projects, while they all benefit from the well-functioning and success of the ecosystem as a whole. This is analogous to natural ecosystems, in which there are specific amounts of raw materials (e.g., light or water) for the species to convert to energy. The species, then, are part of the transfer of this energy (e.g. from the sun to a plant and from a plant to a herbivorous species), while all species benefit from the thriving of the ecosystem.

If we examine the ability of the ecosystem to introduce new actors, we make the separation between host and developing actors. The hosts are involved in the ecosystem when the orchestrators decide to implement an application in their organizational premises, while there is a defined number of (possible) hosts for the ecosystem that includes the regions, hospitals and primary sector practitioners. The ecosystem is not allowing any additional hosts unless the orchestrator decides otherwise. The developing actors, on the other hand, are introduced in the telemedicine ecosystem as part of a project and mainly through a public contest. The management of each project and the orchestrators publish a call for tenders that is typically replied by a number of applicants. Based on a number of predefined criteria, the most suitable tender is chosen from the list of applicants. This is typically a time-demanding procedure and there is an acceptance rate related (typically only one tender accepted, irrelevant the number of applicants). The above characterise the telemedicine ecosystem as a *closed* ecosystem: it is possible for new actors to join but only if the orchestrators allow an opening and after a long and selective procedure.

Examining the platform of the ecosystem, we note that the Danish healthcare includes common technological platforms for secure communication, video communication, and service-oriented computing (the "Danish Healthcare Data Network" (SDN), the "Danish Healthcare Video Hub" (VDX), and the "National Service Platform" (NSP) respectively). These platforms have, however, not been built for purposes of telemedicine (with patients directly involved in use), and thus, we notice the lack of a widely-used, common platform for telemedicine applications. Applications might be using the same systems (like identifying patients through their civil registration number) but the solutions are mainly ad hoc based, which is to some extent the reason for the problems in implementing telemedicine applications.

The nature of the software is also different than many of the known SECOs. Telemedicine applications, in most of the cases, are handling patient data and possible failure of these applications might have consequences in confidentiality, wrong treatment or even prove fatal for patients. This indicates that the applications are of mission-critical nature [11].

Examining the characterization of the telemedicine ecosystem, we note that it deviates from the definition of a SECO in Section 2.1. That is mainly because the technological platforms existing in the telemedicine ecosystem are not telemedical platforms per se. However, we argue for the use of software ecosystem theories in analysing the telemedicine ecosystem for two reasons: (i) the actor structure commits to a SECO actor structure both in terms of symbiotic relationships, business models motivating them and having as products software and services and (ii) we identify that part of the problems in the telemedicine ecosystem is due to the lack of a common and widely used technologic platform. Thus studying the telemedicine ecosystem as a software ecosystem will point to issues needed to improve the software ecosystem for the future.

3. RELATED WORK

In the field of SECO there is a number of papers focusing on actor relationships. The papers by Molder et al. [12] and Jansen et al. [13] study the actor relationships with the ecosystem as a whole by evaluating how open an ecosystem is to new actors. Kabbedijk and Jansen [14] mine the developer activity of the free or open source software (FOSS) Ruby repository, identify three developer roles and conclude with proposals for improving the health of the ecosystem. In their paper, van Angeren et al. [15] analyze the choice of suppliers and supplier-vendor relationships the SECO orchestrators make and identify four supplier strategies. Yu et al. [16] evaluate the collaboration of different FOSS projects in terms of symbiotic relationships.

From the perspective of FOSS, there is an additional number of papers studying the FOSS participants in terms of motivation and collaboration of developers [17, 18], what affects developer activity in a project [19, 20, 21], the organizational structure of participants [22], and tools for extracting information or visualising participant activity [23, 24, 25, 26].

The majority (13 out of 15) of the papers above are focusing on FOSS studies. We recognise a difference in the social structure of a FOSS and proprietary SECO. The actors in a FOSS ecosystem are often single developers or participants, they are many times participating for other reasons than monetary compensation and can, in most of the cases, join a project without great effort (e.g., by creating an account by using a valid e-mail address) [5]. The actors in a non-FOSS ecosystem, such as the telemedicine ecosystem of our study, are mainly companies or public organizations, are objected to harder requirements for joining the ecosystem (e.g., being select in a regional or national call for tenders), and are payed for their work. We argue that these differences

are reflected in the social structure of the ecosystem. To our knowledge, no published study has conducted a quantitative analysis of the actors of a non-FOSS SECO.

4. METHOD

Identifying and analysing the network of actors of the Danish telemedicine ecosystem potentially allows us to characterise the ecosystem as a whole and define its boundary, identify issues that prevent the ecosystem from evolving, and draw conclusions about how it is functioning. Our initial hypotheses are that the telemedicine ecosystem (a) has software systems that are primarily unrelated and (b) the actor structure consists of isolated components. We aim to study the ecosystem by mainly focusing on the following points:

1. Identify the actors of the telemedicine ecosystem and their relationships.
2. Extract information about the ecosystem from the study of the actors and their relationships.

The first point focuses on the identification of the involved parties of the ecosystem, their role, and their relationships among each other. The outcome of this point is a list of actors and connections between the actors. We follow an application-based approach, where we first identify the telemedicine applications and then identify the different actors related to each application. For example, the telemedicine system project KIH is managed by a state-controlled organization (MedCom), implemented in two regions and a number of hospitals and municipalities, and being tested by a university department. These actors are listed as connected with KIH.

To characterise the roles of the actors, we use the roles explained in Section 2.3: "host" that is having an application implemented (e.g., a hospital or municipality) and "developing actor" that is involved in the development or implementation of an application (e.g., a software or medical company). We note that these roles complement the SECO roles mentioned in the SECO background paragraph (2.1). A host can be a niche player, orchestrator (e.g., a municipality) or even an external developer while a developing actor can be a niche player or an external developer. We argue that our own defined roles (host and developing actor) are better suited to draw conclusions about the activity of the specific ecosystem as they are more concrete and help in characterizing the applications.

Our analysis aims at identifying applications from all stages of development, so the collected list of applications can include applications under development, implemented, or discontinued. In this way, we uncover a wider spectrum of actors.

The second point analyses the actors and their interaction and deduct information about the ecosystem. To achieve this, we model the applications, actors and their relationships as a graph and study the properties of this graph with the use of network theory. More specifically, we identify actors according to their level of contribution in the ecosystem. Actors with a high level of contribution to the ecosystem can be either *keystones* or *dominators* [27]. A keystone is an actor that is essential for the ecosystem and potential extinction of a keystone would bring consequences to several actors or the whole ecosystem. The keystone supports the prosperity of the ecosystem through actions that result in the benefit of other actors or the whole ecosystem. A dominator, on the other hand, is similar to a keystone but has the tendency to grow in size by eliminating surrounding species. The roles of the eliminated species in the ecosystem are then either taken over by the dominator or disappear from the ecosystem. The dominators are harmful for the ecosystem's health as they reduce the ecosystem diversity and thus affect niche creation.

In food webs of natural ecosystems, Jordán et al. [28] proposed the keystone index calculation to measure the level to which a species is a keystone in a food chain. In previous work [29], we also used PageRank as a way of measuring the keystone components in the dependency graphs of two OSGi SECOs. PageRank is the algorithm used by Google search engine to rank the results of a search. It assigns a weight for each webpage according to the number of links pointing towards and from this webpage and the weights of the linked pages [30]. In the same work, we compared PageRank with the keystone index and found an overlap of rankings. In related work, Kabbedijk and Jansen [14], used the eigenvector centrality to calculate the importance of a node in a graph of the developer - gem relationship of the Ruby OSS ecosystem. To identify the keystones in the telemedicine ecosystem we use the PageRank algorithm, while we also compare the results to the results from the eigenvector centrality algorithm.

5. ANALYSIS

To evaluate our hypotheses and method, we applied the method to one of the five healthcare regions of Denmark, the Capital Region that is responsible for the Copenhagen area and the Bornholm island. The Capital Region is an appropriate sample for our study as it is the biggest region in terms of population, covering roughly 1.6 million citizens (around 28% of the total population of Denmark) and administering a total of 12 hospitals, 19 psychiatric centers, 247 psychiatrists, 742 GPs, and 164 specialists [7, 31, 32].

We obtained the list of the telemedicine applications and actors from the Capital Region. We verified the list and identified the actors by collecting project-related information (project descriptions, meeting minutes, supplier invoices) from the internet. Our extracted data consist of a total of 28 applications that are running, under development, or discontinued in the Capital Region, 109 actors [1] and 182 connections between actors and applications. These create a network of actors and applications that can be seen on Figure 1[2]. The white nodes are applications, the grey are hosts and the black are developing actors. The size of each node represent the node degree. Since our network is application-based, all the connections are from and towards an application.

We observe that the nodes with higher degree are applications (nine out of the ten nodes with the highest degree). On average one application has ten connections (10.1) out of which 4.72 are developing actors, 4.66 hosts and 0.72 other applications, while one actor is on average connected to a

[1]We note that the actors might also be related to other regions (e.g. an application that is implemented in several hospitals across regions)

[2]A detailed, interactive graph can be found at `http://diku.dk/~kmanikas/medes_2013/`

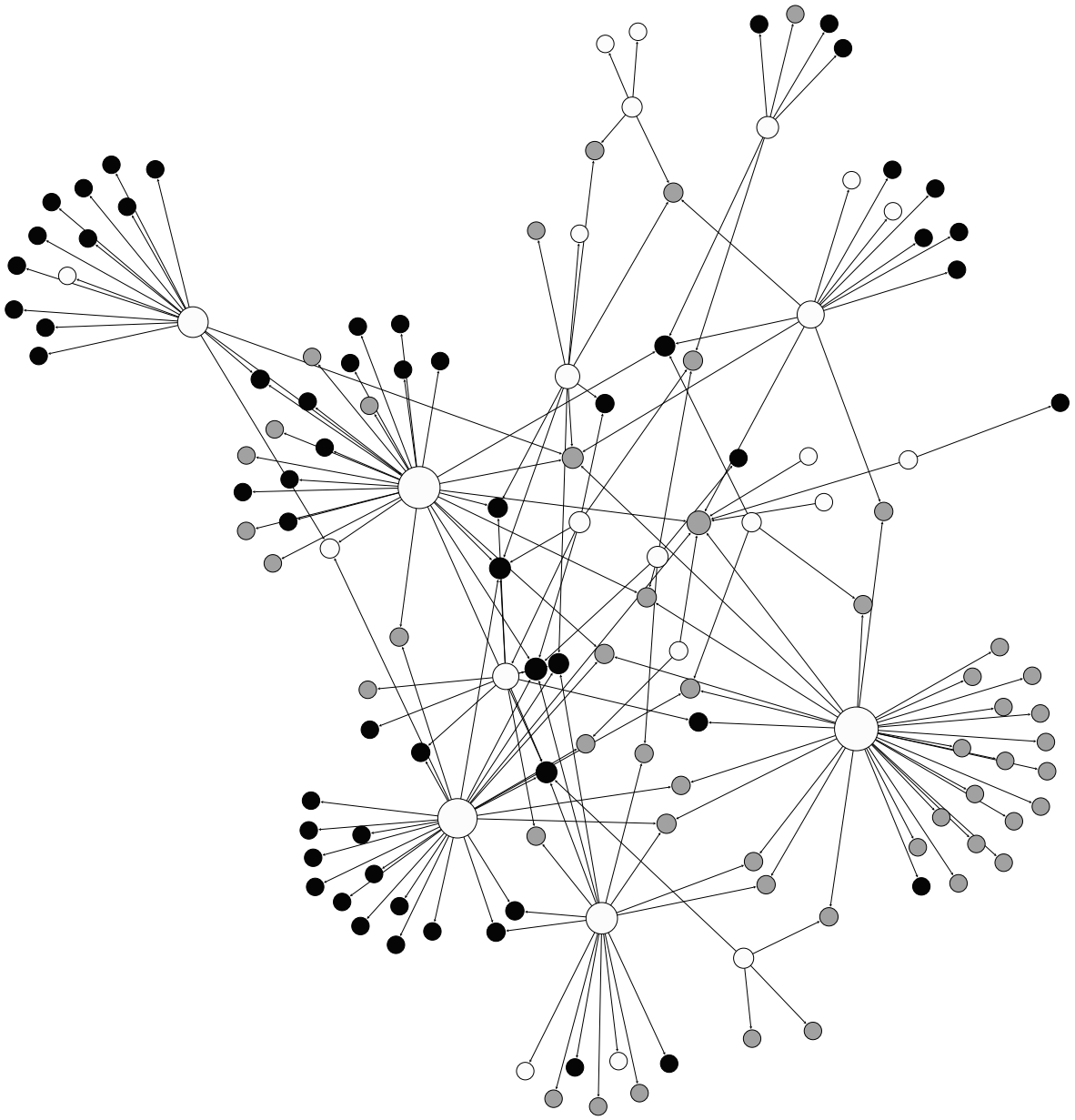


Figure 1: Mapping of the actor and application connections for Capital region. (White nodes: applications, grey: hosts, black: developing actors)

little more than one application (1.52, developing actors: 1.44 and hosts: 1.68). The developing actors connected to the most applications, if we exclude the regions per se[3], are MedCom (5) and the Region's IT, Medico, and Telephony unit (IMT) (4), while the most connected hosts are Hvidovre hospital (8) and Herlev hospital (4). When examining the distribution of the roles, the majority are developing actors (43% of the total nodes), 37% are hosts, while 20% are applications. In analysing this data, we note that the division between the developing actors and hosts is close to equal.

We draw the following conclusions from the above. First, the applications are mainly implemented at one host. There might, however, be a need for more than one host per application implementation, e.g., an application that uses both a hospital and a municipality or an application that handles teleconferences between the doctors of two hospitals. This deviation can explained in at least two ways: (i) there is need for only one implementation, e.g., the application for an implementable cardioverter-defribilator being implemented in a hospital and supporting all the patients with this implant in a country-wide manner. (ii) The applications are host specific and require additional effort to be implemented at a new host, e.g., based on the specific processes of one hospital.

Second, the high number of actors involved in an application is a sign of increased actor diversity. The diversity of actors and their function is arguably a sign of ecosystem health and prosperity [33, 27]. The number of applications per developing actor (1.44) gives the view of applications being developed by specialised developing actors, i.e. that each developing actor has one specific role in only one application of the ecosystem. That can also be explained by the "public call for tenders" actor involvement policy the ecosystem is following: typically only one from a number of applicants is chosen as appropriate for each call. This, as already stated in Section 2.3 characterises the ecosystem as closed in terms of actor involvement. Additionally, it underlines the application interoperability issues the ecosystem is facing. A developing actor is typically involved in only one application without being involved in possible communication or interoperability projects with other applications.

Third, although there are connections between applications (0.72 per application on average), we note that there is no direct connection between the most influential applications (applications with many actors), but the application to application connections are mostly an application connecting to an application with few actors and that being, many times, the only application-to-application connection of both. This supports the view of low interoperability applications.

These conclusions contribute to the view of the telemedicine ecosystem as a group of loosely connected components positioned around the telemedicine applications. Something that can be also noted in Figure 1.

Table 1[4] shows the top ten calculations for PageRank (probability 0.85) and eigenvector centrality. Examining the actors of the PageRank set, we distinguish three actors that have high rankings in both calculations[5]: Hvidovre hospital, IMT and MedCom. Hvidovre hospital appears to be the most influential host actor with participation in eight applications. When looking closer at the activity of the actor, we note that it is involved in the three highest degree applications (applications with most actors) but also to three applications for which Hvidovre hospital is the only host and that seem to be developed specifically for that setting. MedCom is a non-profit, state-established and -funded organization that has the role of creating standards, project management and quality assurance of telemedicine applications. IMT is an organization responsible for the infrastructure of Capital region hospitals with focus on standardised IT, medical technology and telephony.

From the host perspective, the involvement of Hvidovre hospital in projects that are implemented at additional hosts could constitute keystone activity, as it is providing value to the ecosystem by participating in applications that are wider implemented. However, the implementation of applications that are specific to the Hvidovre hospital setting may affect the prosperity of the ecosystem in the long term: other hosts that would require similar applications would either have to spend resources in migrating the existing applications or create new ones. Taking into consideration the financial model of the telemedicine ecosystem, as it looks at the moment, i.e., that there is a fixed amount of funds invested in projects, this kind of activity results in reserving funds that can be used in other ecosystem activity. This implies dominator activity as the actor is taking over funds that could be shared with other hosts.

From the developing actor perspective, we notice that part of the activity of MedCom is to create telemedicine standards that aim at assisting the development of telemedicine applications. That, in combination with the fact that MedCom does not develop applications, implies keystone activity as it provides value to the ecosystem by easing the development of other developing actors. The activity of IMT could be translated as both keystone or dominator. From one side, it is beneficial for the rest of the actors to have a standardised way of communicating to and from the hospitals. On the other hand, however, it could imply dominator activity if IMT apart from standards is also the only supplier of the infrastructure.

When comparing the PageRank and eigenvector centrality sets, we find a small overlap with a Jaccard index[6] of 0.33. However, the most influential host and developing actors (after the exclusion of the regions) appear to be the same with the inversion of IMT and MedCom. Those three actors are also the actors with highest degree, again if we exclude the orchestrator regions. At this point our data are not solid enough to support an argument for or against the use of eigenvector centrality as means of revealing the most influential actors in an ecosystem.

6. DISCUSSION AND FUTURE WORK

Our analysis verified our initial hypotheses of the applications being to a large extent unrelated and the actors being

[3]Our study did not include the orchestrator in the roles, as there were no data to support this role. The regions appear as developing actors.

[4]The full set of calculated metrics can be found on `http://diku.dk/~kmanikas/medes_2013/actors/`

[5]Once more, the regions are not taken into account: In this graph they appear as developing actors as our analysis did not have enough data to support the orchestrator role.

[6]The size of the set intersection divided by size of the set union.

Table 1: Top ten nodes for PageRank and eigenvector centrality calculations.

PageRank		Eigenvector Centrality	
Actor	Type	Actor	Type
Hvidovre Hospital	host	Capital Region	dev
Capital Region	dev	Central Region	dev
IMT	dev	MedCom	dev
Hillerød Hospital	host	Welfare Technology Fund	dev
Amager Hospital	host	Hvidovre Hospital	host
MedCom	dev	Herlev Hospital	host
Central Region	dev	IMT	dev
Ministry of Interior	dev	South Region	dev
Frederikssund Hospital	host	Århus University hospital	host
Rigshospitalet	host	Frederiksberg Hospital	host

host: hosting actor
dev: developing actor

separated into unrelated components. Additionally, with the use of network theory we identified three actors that appear to be influential in the ecosystem and elaborated on their roles. These findings, however interesting for the description of the ecosystem and the verification of our method, are far from complete. In particular, we identify that the regions might not be independent from each other on their activity. This is implied by the existence of hosts from different regions in our actor list. Our next step is to identify the applications and actors from the rest of the regions to compare the overlap and form a complete picture of the actor activity in the ecosystem.

7. CONCLUSION

Telemedicine is a notion that is not as widely applied in the Danish healthcare as one would expect examining the benefits it provides. The benefits of telemedicine applications seem to be counterweighted by, among others, implementation and interoperability issues, silo solutions, and possible lack of use of guidelines and standards. In this paper we focus on the ecosystem structured around the Danish telemedicine services. We describe the telemedicine ecosystem with the state and the five healthcare regions being the orchestrators, having a business model where the actors are publicly funded, and being characterised by the lack of a widely-used common technological infrastructure for mission-critical software. We propose the study of the network of actors and applications to provide better insight to the ecosystem, while, we identify that there is no previous work of this kind in proprietary SECOs. We establish a study method where we define two kinds of actors specific for the telemedicine ecosystem and propose the characterization of the actors and their activity, according to the level of contribution they have to the ecosystem, as keystone or dominator. We apply our method to the Capital Region of Denmark, the most populated of the five healthcare regions, and (i) identify the tendency of single implementation applications, either because of the nature of the applications or because of the applications being ad-hoc solutions, (ii) identify variability in actors and their functions, (iii) characterise the ecosystem as rather closed in terms of introducing new actors, (iv) identify the actor structure clustered around the applications with the clusters separated from each other, (v) identify three actors as the most influential, and (vi) characterise the activity of these actors as keystone and dominator.

Acknowledgements

We would like to thank Judith Lørup Rindum from the Capital Region for providing us with the list of telemedicine applications. This work has been partially funded by the Connect2Care project[7].

8. REFERENCES

[1] R. Wootton. Recent advances: Telemedicine. *British Medical Journal*, 323(7312):557–560, September, 2001.

[2] K.M. Hansen, M. Ingstrup, M. Kyng, and J.W. Olsen. Towards a software ecosystem of healthcare services. In *Infrastructures for Healthcare: Global Healthcare, Proceedings of the 3rd International Workshop*, pages 28–31, Copenhagen, Denmark, June, 2011.

[3] A. Steventon, M. Bardsley, J. Billings, J. Dixon, H. Doll, S. Hirani, M. Cartwright, L. Rixon, M. Knapp, C. Henderson, et al. Effect of telehealth on use of secondary care and mortality: findings from the whole system demonstrator cluster randomised trial. *BMJ: British Medical Journal*, 344, 2012.

[4] H. B. Christensen, M. Christensen, K. M. Hansen, K. Manikas, and S. Urazimbetova. Requirements for a Software-Intensive Ecosystem for Telemedicine. In *Med@Tel 2012: Global Telemedicine and eHealth Updates*, volume 5, pages 423–427, Luxembourg, April, 2012.

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

[6] Danish Ministry of Health and Prevention. http://www.sum.dk/Aktuelt/Publikationer/Publikationer/UK_Healthcare_in_DK.aspx. Health care in denmark, 2008.

[7] The Danish Ministry of the Interior and Health. The local government reform - in brief, 2008.

[8] G. Esping-Andersen. *The three worlds of welfare capitalism*, volume 6. Polity press Cambridge, 1990.

[9] P. T. Andersen and J. J. Jensen. Healthcare reform in denmark. *Scandinavian journal of public health*, 38(3):246–252, 2010.

[10] World Health Organization. Telemedicine. opportunities and developments in member states. Global Observatory for eHealth series – Volume 2, 2010.

[7] `http://www.partnerskabetunik.dk/projekter/connect2care.aspx`

[11] H.B. Christensen and K.M. Hansen. Net4care: Towards a mission-critical software ecosystem. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, pages 224–228. IEEE, Helsinki, Finland, August, 2012.

[12] J. te Molder, B. van Lier, and S. Jansen. Clopenness of systems: The interwoven nature of ecosystems. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 52–64. CEUR-WS, Brussels, Belgium, June, 2011.

[13] S. Jansen, S. Brinkkemper, J. Souer, and L. 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.

[14] J. Kabbedijk and S. Jansen. Steering insight: An exploration of the ruby software ecosystem. In *Software Business*, volume 80 of *Lecture Notes in Business Information Processing*, pages 44–55. Springer Berlin Heidelberg, 2011. 10.1007/978-3-642-21544-5_5.

[15] J. van Angeren, V. Blijleven, and S. Jansen. Relationship intimacy in software ecosystems: a survey of the dutch software industry. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 68–75, California, USA, November, 2011.

[16] L. Yu, S. Ramaswamy, and J. Bush. Software evolvability: An ecosystem point of view. In *Software Evolvability, 2007 Third International IEEE Workshop on*, pages 75 –80, Paris, France, October, 2007.

[17] W. Scacchi. Collaboration practices and affordances in free/open source software development. In *Collaborative Software Engineering*, pages 307–327. Springer Berlin Heidelberg, 2010. 10.1007/978-3-642-10294-3_15.

[18] T. Mens and M. Goeminne. Analysing the evolution of social aspects of open source software ecosystems. In *Third International Workshop on Software Ecosystems (IWSECO-2011)*, pages 1–14. CEUR-WS, Brussels, Belgium, June, 2011.

[19] E. Ververs, R. van Bommel, and S. Jansen. Influences on developer participation in the Debian software ecosystem. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 89–93, California, USA, November, 2011.

[20] K. van Ingen, J. van Ommen, and S. Jansen. Improving activity in communities of practice through software release management. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '11, pages 94–98, California, USA, November, 2011.

[21] R. P. M. Krishna and K. G. Srinivasa. Analysis of projects and volunteer participation in large scale free and open source software ecosystem. *SIGSOFT Softw. Eng. Notes*, 36:1–5, March, 2011.

[22] C. Jergensen, A. Sarma, and P. Wagstrom. The onion patch: migration in open source ecosystems. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, ESEC/FSE '11, pages 70–80, Szeged, Hungary, September, 2011.

[23] M. Lungu, M. Lanza, T. Gîrba, and R. Robbes. The small project observatory: Visualizing software ecosystems. *Science of Computer Programming*, 75(4):264 – 275, 2010. Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008).

[24] M. Lungu, J. Malnati, and M. Lanza. Visualizing gnome with the small project observatory. In *Mining Software Repositories, 2009. MSR '09. 6th IEEE International Working Conference on*, pages 103 –106, Vancouver, Canada, May, 2009.

[25] M. Goeminne and T. Mens. A framework for analysing and visualising open source software ecosystems. In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, IWPSE-EVOL '10, pages 42–47, Antwerp, Belgium, September, 2010.

[26] S. Neu, M. Lanza, L. Hattori, and M. D'Ambros. Telling stories about gnome with complicity. In *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*, pages 1 –8, Virginia, USA, September 2011.

[27] M. Iansiti and R. Levien. Keystones and dominators: Framing operating and technology strategy in a business ecosystem. *Harvard Business School, Boston*, 2004.

[28] F. Jordán, A. Takács-Sánta, and I. Molnár. A reliability theoretical quest for keystones. *Oikos*, pages 453–462, 1999.

[29] K. M. Hansen and K. Manikas. Towards a network ecology of software ecosystems: an analysis of two OSGi ecosystems. In *Proceedings of the 25th International Conference on Software Engineering & Knowledge Engineering (SEKE'2013)*, Boston, USA, June, 2013.

[30] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[31] The Danish e Health Portal. https://www.sundhed.dk/service/english/. Accessed: January 2013.

[32] The Capital Region of Denmark. http://www.regionh.dk/english/english.htm. Accessed: January 2013.

[33] D. Rapport, R. Costanza, A. McMichael. Assessing ecosystem health. *Trends in Ecology & Evolution*, 13(10): 397–402, 1998.

Paper 6

Analysis and Design of Software Ecosystem Architectures – Towards the 4S Telemedicine Ecosystem

Christensen, H. B., Hansen, K. M., Kyng, M., and Manikas, K. (2014). Analysis and design of software ecosystem architectures – towards the 4s telemedicine ecosystem. *Information and Software Technology*, 56(11):1476 – 1492

Information and Software Technology 56 (2014) 1476–1492

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Analysis and design of software ecosystem architectures – Towards the 4S telemedicine ecosystem

Henrik Bærbak Christensen [a], Klaus Marius Hansen [b,*], Morten Kyng [a,c], Konstantinos Manikas [b]

[a] Department of Computer Science, Aarhus University, Denmark
[b] Department of Computer Science (DIKU), University of Copenhagen, Denmark
[c] The Alexandra Institute, Aarhus, Denmark

ARTICLE INFO

Article history:
Received 4 July 2013
Received in revised form 8 May 2014
Accepted 9 May 2014
Available online 28 May 2014

Keywords:
Software ecosystem architecture
Third-party sponsored software ecosystems
Telemedicine software ecosystems

ABSTRACT

Context: Telemedicine, the provision of health care at a distance, is arguably an effective way of increasing access to, reducing cost of, and improving quality of care. However, the deployment of telemedicine is faced with standards that are hard to use, application-specific data models, and application stove-pipes that inhibit the adoption of telemedical solutions. To which extent can a software ecosystem approach to telemedicine alleviate this?
Objective: In this article, we define the concept of *software ecosystem architecture* as the structure(s) of a software ecosystem comprising elements, relations among them, and properties of both. Our objective is to show how this concept can be used (i) in the analysis of existing software ecosystems and (ii) in the design of new software ecosystems.
Method: We performed a mixed-method study that consisted of a case study and an experiment. For (i), we performed a descriptive, revelatory case study of the Danish telemedicine ecosystem and for (ii), we experimentally designed, implemented, and evaluated the architecture of 4S.
Results: We contribute in three areas. First, we define the software ecosystem architecture concept that captures organization, business, and software aspects of software ecosystems. Secondly, we apply this concept in our case study and demonstrate that it is a viable concept for software ecosystem analysis. Finally, based on our experiments, we discuss the practice of software engineering for software ecosystems drawn from experience in creating and evolving the 4S telemedicine ecosystem.
Conclusion: The concept of software ecosystem architecture can be used analytically and constructively in respectively the analysis and design of software ecosystems.

1. Introduction

The research field of *software ecosystems* has emerged as the study of the complex interaction between extensible software frameworks and software architecture(s) on one hand, and organizations, users, customers, developers, and businesses on the other. It is inspired by natural ecosystems in which species are characterized by symbiotic relationships and their survival relies heavily on the survival of the ecosystem [45,33,6,10,11,39]. We define a 'software ecosystem' as:

> the interaction of a set of actors on top of a common technological platform that results in a number of software solutions or services. [42].

Further, "Each actor is motivated by a set of interests or business models and connected to the rest of the actors and the ecosystem as a whole with symbiotic relationships, while, the technological platform is structured in a way that allows the involvement and contribution of the different actors." [42].

A well-known example of a software ecosystem is the Android ecosystem. From a software ecosystem point of view, Google controls the Android platform while external developers can build applications ("apps") that are distributed to Android users via the Google Play store. Thus, Google has collaborated with external developers to quickly build functionality in the form of more than 700,000 apps [64]. In this way, the Android software ecosystem has arguably helped Google increase the value of Android for its users, increased attractiveness, accelerated innovation, and decreased cost [6].

We may distinguish between two main elements of software ecosystems:

* Corresponding author. Tel.: +45 61732721.
E-mail addresses: hbc@cs.au.dk (H.B. Christensen), klausmh@diku.dk (K.M. Hansen), mkyng@cs.au.dk (M. Kyng), kmanikas@diku.dk (K. Manikas).

http://dx.doi.org/10.1016/j.infsof.2014.05.002
0950-5849/

- *Actors* that may, e.g., be individuals or organizations. Depending on their activities in the ecosystem, they can have different roles including 'orchestrator' (such as Google in the Android ecosystem), 'keystone' (such as Samsung in the Google ecosystem), and 'niche player' (such as external software developers in the Android ecosystem). Additionally, each actor has an incentive for being active in the ecosystem that, often, can be represented by a business model.
- *Software* that exists as a platform/framework or as software solutions or services built on the platform. Software forms a main element of software ecosystems and a software ecosystem can be studied through the software elements that it consists of, their relations, and properties. The ability of the platform to incorporate different elements and the interaction and interoperability of the elements, are characteristics of the platform.

In this article we report on work in relation to telemedicine ecosystems spanning the period from 2008 to the present. It covers analysis of the general Danish telemedicine ecosystem, work on a new technological infrastructure platform, named Net4Care [47], and design as well as partial evaluation of a new organization created to accelerate the evolution of the ecosystem based on the Net4Care platform. The new organization is called "Stiftelsen for Softwarebaserede SundhedsServices" (in English: The Foundation for Software-based Healthcare Services), and hereafter referred to by its acronym "4S".

The 4S organization must handle a large set of diverse stakeholders, including national healthcare agencies, regional hospitals, software development houses, IT departments, etc. with highly complex interactions between. To understand and manage this complexity, we propose the concept of *software ecosystem architecture*, extending previous work by Manikas and Hansen [42]. We use this concept to frame a case study of the current Danish telemedicine ecosystem to inform the creation of 4S. In this context, the concept provides a common terminology across the central three structures of a software ecosystem: the organizational structure, the business structure, and the software structure.

We investigate this concept to ultimately answer the following research question:

> *"How can software ecosystems be modeled in a systematic way that allows reasoning about software ecosystems while details of the software ecosystem can be abstracted away?"*

Section 3 discusses the research question in detail.

Our contributions are threefold. First, we define and discuss the *software ecosystem architecture* concept. Secondly, we present the case of the current Danish telemedicine ecosystem in terms of the proposed concept, and discuss challenges that are relevant in areas beyond telemedicine. Finally, we present how the practice of software engineering is affected, through describing the creation and evolution of a central ecosystem architecture, Net4Care, that serves as a reference architecture and learning vehicle for telemedicine for the actors in 4S.

2. Related work

In this article, we analyze software ecosystems using the concept of 'software ecosystem architecture' and report on the case of the Danish telemedicine ecosystem. To our knowledge there is no previous work on software ecosystems for telemedicine or even for healthcare as such, apart from our own previous publications [24,15,16,40]. However, there is significant work on conceptualizing and modeling software ecosystems.

For example, there is related work that has influenced the way software ecosystems are modeled either by contributing to conceptualization [55,1,2,13] or by addressing a specific aspect of software ecosystems such as the platform and software component architecture [7,9,8,53,34].

Jansen et al. [33,31] and Boucharas et al. [12] propose the analysis and modeling of a software ecosystem from the software vendor perspective and separate ecosystems in three levels: the software ecosystem, the software supply network and the software vendor level. Jansen et al. [30], similarly, define three scope levels for software ecosystems: an external view, an internal view and an organization centric-perspective. Bosch [6] categorize software ecosystems according to their platform as operating system-centric, application-centric, or end-user programming-centric. Campbell and Ahmed [14] identify three dimensions of the engineering process of software ecosystems: business, architectural, and social. These map to our business, software, and organizational structure (with differences in the social dimension versus what the organizational structure covers). While their focus is on the engineering process, our focus is on the structure of an underlying software ecosystem.

Additionally, there is significant related work on quality aspects of software ecosystems. One such aspect is 'health' or 'sustainability'. In this context, van den Berk et al. [61] propose an ecosystem-based model for assessing the strategy of a software ecosystem called SECO-SAM. In their paper, they make an analogy between human health and ecosystem health and model the ecosystem health as being influenced by the biology of the ecosystem, the lifestyle, the environment, and the intervention of so-called healthcare organizations. Jansen et al. [30] define ecosystem health as a characteristic of the software supply network level in their three-level model mentioned above. Additionally, they propose the application of the measures of den Hartigh et al. [19] for defining the health of software ecosystems. van Angeren et al. [60] describe the robustness of the Iansiti and Levien [27] health measures of business ecosystems as an important factor for vendors that choose to depend on a software ecosystem. McGregor [43] translates the measures by Iansiti and Levien [27] to measures that can be applied to open source projects, Kilamo et al. [35,36] propose a framework for going from a proprietary to a Free/Libre/Open Source Software (FLOSS) ecosystem. One of the framework activities is setting up a "community watchdog" to assess three aspects of the newly created ecosystem: the community, the software, and "how well the objectives of the company are met". Although not directly stated, the watchdog indirectly assesses the health, while provide a number of measures to be applied in FLOSS ecosystems. Manikas and Hansen [41] propose a framework for the measurement of the ecosystem health. This framework consists of three ecosystem aspects that influence the ecosystem health: the actors, software and orchestration. In our work, we seek to understand how the structure of software ecosystems can be used to reason about these aspect also eventually with respect to health.

3. Method

We base our research on our own systematic literature review [42] and existing mapping studies and literature analyzes [1,55] to provide input and formulate our research question. The main research of this work can be summarized by the question:

> *"How can software ecosystems be modeled in a systematic way that allows reasoning about software ecosystems while details of the software ecosystem can be abstracted away?"*

More specifically, in this article, we are concerned with the following sub questions:

"(a) How can the concept of 'software ecosystem architecture' be used to model an existing (Danish telemedicine) software ecosystem to find areas of improvement?"

"(b) How can the concept of 'software ecosystem architecture' support the engineering of a new (Danish telemedicine) software ecosystem?"

In this article, we introduce the concept of *software ecosystem architecture* (explained in Section 4) and propose to model software ecosystems through their architecture. We apply our concept in a mixed method study: a descriptive case study in the Danish telemedicine ecosystem and an experiment in the Danish telemedicine ecosystem by creating the 4S organization including the Net4Care technological platform.

We use the Danish telemedicine ecosystem as our case study unit of analysis because we have access to information that can provide a deep insight on the case making it what Yin [67] refers to as a *revelatory* case study.

Our experiment focuses on the design of software ecosystems. We apply the concept of software ecosystem architecture and engineer a technological platform, Net4Care, investigating how a software framework can be engineered so that it can serve as a common technological platform, create the 4S organization to serve as the orchestrator of the platform, and change the business model of the telemedicine ecosystem.

Table 1 shows the data collection methods for the case study and the data created from our experiment. For both, the telemedicine ecosystem is analyzed in the three structures of the ecosystem architecture: organizational, software and business structure.

In order to describe the organizational structure of our case study, we collected data with qualitative methods, e.g., interviews/discussions with ecosystem orchestrators actors, external participants and interested parties from public authorities and organizations as well as companies. The public authorities included The National eHealth Authority, where we had several meetings with head of section, software architects and people responsible for standards: MedCom, The Danish Healthcare Datanetwork, where we held meetings with the deputy head and the person in charge of international projects; the five Danish regions, where we had several meetings with managers of telemedicine projects, people heading telemedicine centers, IT departments and departments for procurement and medical technology; and a small number of municipalities where we had meetings with managers of telemedicine projects and managers of departments responsible for telemedicine. From companies we had meetings with CEOs and telemedicine managers from providers of telemedicine, electronic patient records, and care records. The companies involved are Capgemini (now Capgemini Sogeti), CSC Scandihealth, KMD, Logica (now CGI), Systematic, SilverBullet, Sekoia, TDC, and Trifork.

In addition we studied publicly available records form several telemedicine projects, including the two largest projects from the National action plan for deployment of telemedicine. The records included meeting minutes, project reports and project participants.

Finally, we conducted a quantitative analysis of the network of actors and telemedicine application of a part (the largest of the five healthcare regions, the Capital Region, in Denmark) of the ecosystem [40].

Similarly for the characterization of the software structure of the ecosystem: we reviewed a number of existing applications and systems (FMK [22], RRS [54], TELEKAT [57], VAGUS, Telesår[58], EgenJournal [21], Sundt Hjem) and standards ([26], CDA ([5]), PHMR [52], XDS.b [66]), used the application network in the actor – application qualitative study mentioned above and interviewed an SMB in the field of telemedical application development Viewcare [62].

The variability of sources in the organizational structure and software structure of the case study addresses source triangulation [51,20]. Additionally, in this article we are using a mixed method of a case study and an experiment. Therefore, we obtain method triangulation [51,20] in the characterization of the software ecosystem architecture concept.

4. Software ecosystem architecture

We define the concept of 'software ecosystem architecture' by generalizing the definition by Bass et al. [4] of 'software architecture' (and extending on the definition of Manikas and Hansen [42]):

> The architecture of a software ecosystem is the set of structures needed to reason about the software ecosystem, which comprise actor and software elements, relations among them, and their properties.

The definition stresses that the architecture of a software ecosystem consists of multiple structures, each consisting of actor and software elements. Software forms the core of a software ecosystem, therefore the *software structure* of the software ecosystem is important. Moreover, a main purpose of software ecosystem actors is to create value (in a for-profit or non-profit manner) and thus the *business structure* of a software ecosystem becomes relevant. Finally, it is important to govern the interaction and organization of actors and software (e.g., for an actor to provide a software-based service in the ecosystem) and thus the *organizational structure* of a software ecosystem becomes important.

While many other structures can be discerned for software ecosystems, we argue that the above three are highly relevant and critical for reasoning about a software ecosystem. Consequently, we discuss these in turn and apply them in our analysis and design. Table 2 summarizes this discussion.

4.1. Organizational structure

The organizational structure of a software ecosystem contains actor and software elements that are related to the governance of the interaction and organization of the elements in the ecosystem. Important aspects of the organizational structure are the sets of actor and software elements included, the boundary of the

Table 1
Data collected and created during our case study and experiment.

Danish telemedicine ecosystem		Experiment: 4S	
Unit of analysis	Data collection	Unit of design	Data creation
Organizational structure	Interviews, publicly available records, qualitative analysis of ecosystem actor-application network	Organizational structure	4S organization creation
Business structure	Archival records of projects including Telesår	Business structure	Business modeling
Software structure	Interview with SMBs; analysis of existing applications, technologies, & standards	Software structure	Net4Care platform development & evaluation

Table 2
Examples of structures of software ecosystems.

Structure	Elements	Relations	Models
Organizational structure	Applications, platform, orchestrator, users, developers, boards, development projects, plans	Governed by, developed by, maintained by, connected to	Organizational structure, organization relationship & interaction, organization roles
Business structure	Products, services, partners, customers, resources	Channels, customer relationship, revenue stream	Business model canvas
Software structure	Modules, functions, services, nodes, developers	Depends on, used by, deployed on, developed by	Software architecture description

ecosystem they define, and how the structure governs interactions and support coordination among actors and software elements.

The interaction of actors is also related to the role each actor serves in the ecosystem. Specific roles might be more prone or even necessary to have interaction in different kinds of ecosystems, e.g., software developing organizations would have to interact with certification organizations in an ecosystem to promote their software products.

Moreover, the number of actors involved in an ecosystem and the level of selection they have to go through to be involved is part of the organizational structure. This is described by assessing how open the ecosystem is to external actors (e.g., by using the approach of Jansen et al. [32] or Manikas and Hansen [40]). Finally, the actors involved in the ecosystem usually have a commitment to the ecosystem. This commitment makes them aligned with the common goal of the ecosystem: its sustainability. Each actor has its own set of goals, however in order for the individual actors goals to be achieved and continued, the survival and thriving of the ecosystem is usually required. Exceptions are e.g. the cases where an actor decides to favor another ecosystem or where conflicts among actors weaken the sustainability of an ecosystem.

4.2. Business structure

The business structure contains actor and software elements that are related to how actors create, deliver, and capture value. 'Value' here refers to the benefit an actor gets from the software ecosystem, e.g., in form of need satisfaction or problem solution. A software element may deliver value in itself (e.g., an application) or be a resource in creating value (e.g., a platform or a reusable component/service). This structure is important in reasoning about cost, revenue, and/or sustainability of the software ecosystem. Note that business models do not necessarily model commercial organizations' businesses.

We describe business structure through business models using the 'business model ontology' formalism [49,50]. 'Business models' are here defined as

> A business model describes the rationale of how an organization creates, delivers, and captures value.

An excerpt of the business model ontology is shown in Fig. 1. Four aspects of business models are distinguished: the 'product', 'customer interface', 'infrastructure management', and 'financial' aspects.

In our models, we use the practical realization of Osterwalder's business ontology as 'business model canvases'. Here, the product aspects are covered by the 'value proposition' building block that concerns what values an organization is providing for a customer. The customer interface aspects are covered by the 'customer segment', 'channel', and 'customer relationship' building blocks that respectively describe types of customers, how customers are reached, and the type of relationships that are established to customers. The infrastructure management aspect is managed by the 'key partners', 'key resources', and 'key activities' building blocks. Key resources are the most important assets (e.g., physical or intellectual) whereas key activities are the most import actions (e.g., development or platform management) that are required for the business model to work. Finally, the financial aspects are covered by the 'cost structure' and 'revenue stream' building blocks that describe expense and income respectively in the business model.

4.3. Software structure

The software structure contains actor and software elements that are related to the production of applications in the software ecosystem. The primary actors are developers of the software ecosystem platform and of applications. Software elements may (recursively) be seen as consisting of multiple structures such as units of code (i.e., modules), runtime functions (i.e., components),

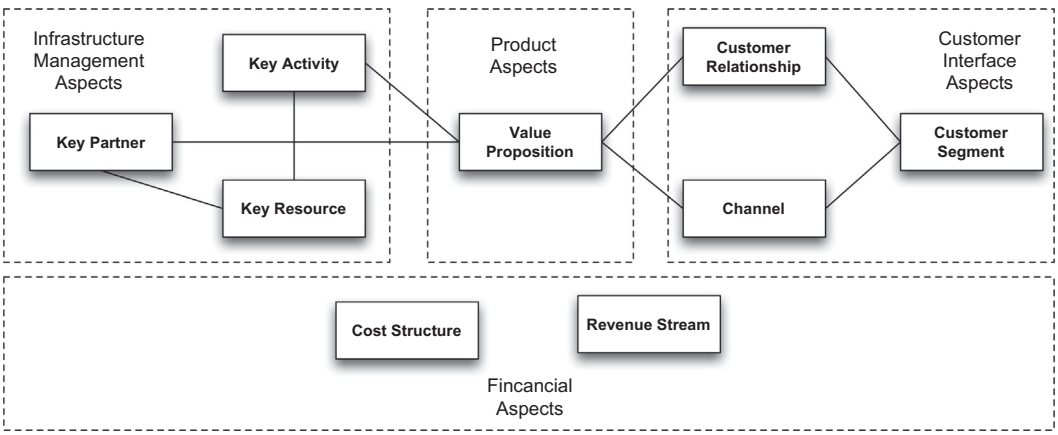


Fig. 1. Business model ontology taken from Osterwalder [49] with business model canvas terminology added.

or deployment nodes. These structures are important in reasoning about system quality attributes such as modifiability, performance, availability, and security [4]. Software architecture (models) are suitable for this type of reasoning and thus we model the software structure of software ecosystem architectures through software architecture descriptions.

We describe software structure in alignment with the ISO/IEC 42010 standard [28], see Fig. 2. Here an 'architecture description' consists of a set of 'architectural views' each of which adheres to the convention of an 'architectural viewpoint'. An example of an architectural description that adheres to this would be a 4 + 1 views-based architectural description [38]. Furthermore architecture decisions are described through an 'architecture rationale' and relations between elements in an architectural description are modeled through 'correspondences'. Architectural decisions are made to support 'architectural requirements'. An architectural view embodies 'architectural models' often in the form of (UML) diagrams. A view models one or more structures through its architectural models.

In our architectural descriptions, e.g., we use a set of viewpoints that, using UML, model how software is developed (through a "development view"), how software behaves at runtime (through a "functional view"), and how software is deployed to hardware (through a "deployment view") [25]. We use *quality attribute scenarios*, as defined by [4], to describe architectural requirements.

A example functional model for Net4Care (which will be introduced in Section 6) is shown in Fig. 8 on page 27, and examples of quality attribute scenarios are shown in Section 6.3.

4.4. Relationships among structures

The main relationships among the three structures that we emphasize are shown in Fig. 3. In addition numerous other relations exist. For example, a set of software services that are part of the software structure may provide the basis for the value creation of a business as described in the business structure and be created and governed as described in the organizational structure.

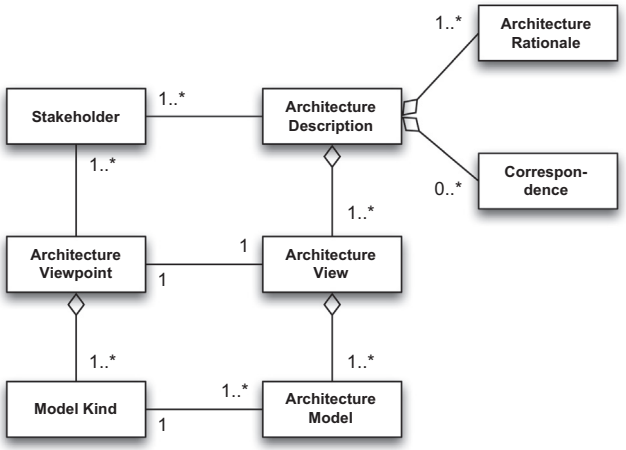


Fig. 2. Excerpt of the ISO/IEC 42010 architecture description ontology.

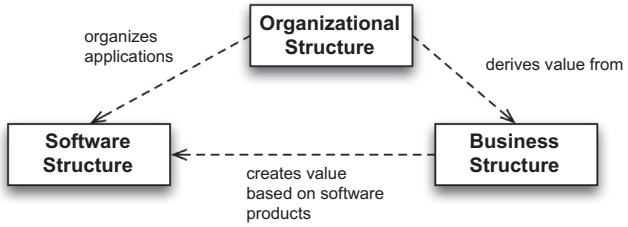


Fig. 3. Main relationships between structures.

5. Danish telemedicine software ecosystem architecture analysis

This study focuses on *telemedicine* as an application domain for software ecosystems in the Danish healthcare. The World Health Organization (WHO) defines telemedicine as

> The delivery of health care services, where distance is a critical factor, by all health care professionals using information and communication technologies for the exchange of valid information for diagnosis, treatment and prevention of disease and injuries, research and evaluation, and for the continuing education of health care providers, all in the interests of advancing the health of individuals and their communities [65].

To this extent, the unit of analysis of our case study is defined by the ecosystem around the telemedicine services in Danish healthcare. In this section, we describe our case study using the concept of software ecosystem architecture, described in Section 4. The aim is to find areas of improvements, cf. question (a) in Section 3. How we address the areas identified is described in Section 6.

Below, we first analyze the organizational structure of telemedicine in Denmark, the government policies developed to address need of increased uptake, and the issues the policy is intended to address. Next, we provide a more detailed analysis of current issues based on the business and software structures of the software ecosystem architecture. Finally, we outline the challenges remaining in the ecosystem under study.

5.1. Organizational structure

In Denmark, telemedicine services form part of the healthcare services offered by the public healthcare system. The telemedicine ecosystem is administered by the same administrative organs as the general healthcare. The organization of the healthcare system is fairly decentralised and has three administrative levels [48]:

State level The Ministry of Health governs the regional and municipal organization and management of healthcare. This includes organizations dealing with national infrastructure standards, architecture, compliance testing and implementation, and organizations dealing with cost structure and value creation.

Regional level Five geographical regions each own and run hospitals and finance private practitioners (in particular general practitioners (GPs)). Financing is based on transfers from the state and municipal levels. This includes organizations dealing with regional hospital systems: architecture, tenders and requirements, operation and national reporting and organizations dealing with general practitioners.

Local level There are 98 municipalities that are locally responsible for prevention, health promotion and rehabilitation. Furthermore, the local level is also responsible for health-related services such as home care and care in nursing homes. This includes organizations dealing with municipality healthcare: architecture principles, tenders and requirements, and operation.

A main issue is the relations among the different elements listed above, and the kind of coordination and predictability that these relations support. As a result of this, telemedicine in Denmark has been characterized by hundreds of uncoordinated, small projects, each developing their own solution, including infrastructure. Those solutions are not able to share data, due to the lack of common infrastructure, and they most often disappear when the resources of the project are consumed. This has resulted in more than 350 current telemedicine initiatives [44] of which the minority are in production.

This is also supported by our qualitative study of telemedicine applications and the organizations [40]. We analyzed the telemedicine applications and organizations for the Capital Region of Denmark. This is the largest, in terms of population, of the five healthcare regions in Denmark, with around 30% of the country's population. We identified the organizations related to the telemedicine applications implemented in the region and mapped the relationships between organizations, between organizations and applications, and between applications and applications. Our results revealed an ecosystem clustered around the telemedicine applications with low connectivity between the clusters. In other words, we noted a low application interaction and the tendency of the organizations to be connected to mainly one telemedicine application.

In order to examine how open the ecosystem is to external organizations, we separate the organizations in three roles that are important in the telemedicine ecosystem[1]:

- *Developing or external organization*. Organizations that are involved in the ecosystem with a specific task. This can include the development, maintenance or support of an application, the project management, supplier of related assets or services. The involvement of developing organizations is done, as we discuss in Section 5.2, either with a call for tenders where the actor with the accepted tender is appointed for the required task, or for services with cost lower than 500,000 Danish crowns (about 65,000 euros) direct fee-specific contracting. The ecosystem is relatively closed to external developing organizations: the ecosystem allows new developing organizations to be involved but new organizations are subjected to an acceptance rate (usually one out the applicants) and following a procedure of submitting a tender that might prove time and resource demanding. While direct contracting is sometimes allowed, this only includes minor tasks and the involvement is time-limited.
- *Host*. Organizations that are hosting telemedicine applications in their organizational premisses. This kind of organizations typically represent the product owner and customer/end user. The ecosystem is closed to new host organizations: There is currently a number of hospitals and municipalities that can be used as hosts in telemedical projects. Orchestrators can decide to include other organizations as hosts if there is a need. Moreover, in some cases, as we also discuss in Section 5.3, a developing organization might also serve as an application host. In that case, the ecosystem is almost as open to hosts of this kind as to developing organizations, but with the additional restriction that the host should commit to privacy and security regulations concerning healthcare data.[2]
- *Orchestrator*. Organizations involved in the governing body of the ecosystem typically responsible for the technological platform(s). As the ecosystem does not have one common technological platform, the assessment of how open the ecosystem is to organizations of this role is not possible. Orchestrators of existing platforms that are not ecosystem-wide (e.g. National Service Platform and Healthcare Datanet) have been introduced by appointment of state level organizations.

5.2. Business structure

Telemedicine systems, in a Danish context, are typically developed as part of a *project*. One or several orchestrators decide on investing in solving a specific problem. A project is then initiated where service provider actors (developing organizations) might be involved. The required activities and processes are identified and, if external actors are needed, a public call for tenders is announced. External actors are selected based on their proposals for large projects (cf. Section 5.1).

As an example, consider the national telemedicine project "Telesår".[3] The project aims at bringing expert diagnosis and treatment of ulcers to patients through the use of a mobile phone with a camera and a web-based electronic ulcer journal. The rationale behind this project is that expert ulcer diagnosis for patients with limited mobility (e.g., elderly) is expensive but at the same time necessary as these patients are in the high risk of developing severe ulcer complications often resulting in amputations. A usage scenario is that a home-care nurse visits an elderly patient with a diabetic foot ulcer and – in connection with changing the bandage – takes a picture with the mobile phone and then uploads it in the "ulcer journal". If the nurse finds that action might be needed immediately, she may set up an on-line conference with a dermatologist who, at the backend, logs in to the ulcer journal from his or her office and analyzes the picture of the patient, evaluating how the patient needs to be treated. If no immediate action is deemed necessary the dermatologist evaluates the pictures off-line.

In this scenario, the patient receives better diagnosis and treatment with fewer visits to the specialized hospital departments. The dermatologist can do a better job through closer observation and a specialized record. Furthermore the specialist may treat more patients. Also the home-care nurse learns from the dialog with the dermatologist and is thus able to provide the patient with better care. The regions and municipalities get more value for money, since faster diagnosis and earlier treatment may be provided, while the state increases the efficiency of the provided healthcare services by reducing costs.

A business model canvas for the "Telesår" project is shown in Fig. 4. The history of the project and the business model illustrates some of the issues with the current Danish telemedicine ecosystem. First, projects related to telemedical ulcer treatments have a history that goes back approximately 10 years as local (medical) research projects [17]. Secondly, the project, while now being implemented on a national scale as one of the five projects mentioned in Section 5.4, uses a proprietary electronic journal ("pleje.net") that to our knowledge does not integrate with national or international standards and that as such does not reuse national services (see Section 5.3).

5.3. Software structure

In 2010, we conducted a study of the technical status of current telemedicine projects (RRS, TELEKAT, Telesår, EgenJournal), a commercial system (ViewCare), and international experience and standards (HL7, CDA, and PHMR).

The study included both actors and software architecture elements and relationships, and included interviews with regional hospital IT departments, architects and developers, documentation

[1] A similar classification and evaluation was followed in Manikas and Hansen [40].

[2] Traditional hosts like hospitals are also committed to this kinds of regulations, but this does not appear as cumbersome as this is part of the everyday work in a hospital.

[3] http://medcom.dk/wm112455.

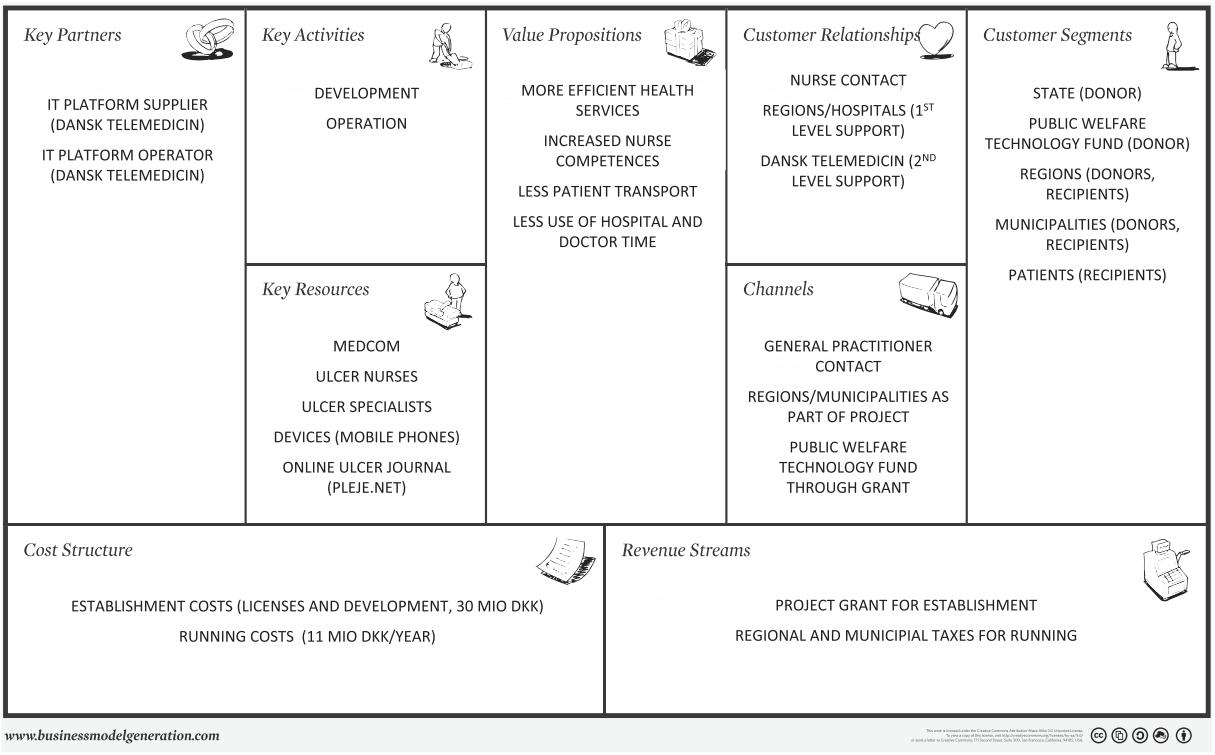


Fig. 4. Telesår telemedicine project organization business model canvas.

reviews, and the development of architectural functional and deployment views for the studied systems. Based on this raw material, a commonality/variability analysis was conducted.

Perhaps not surprisingly as all studied projects and products have the same core use case: "Measure clinical information in the home of a patient, send it to a hospital server for storage and review by a clinician", the same software architecture was shared between all software systems. They were all variants of a three-tier information systems, as depicted in Fig. 5. That is, a monitoring application deployed in the home collects various measurements and uploads them to a proprietary project server, stores data in a proprietary format in a proprietary database, and allows clinicians to browse and view data using a clinician application. Essentially all projects were "stove-pipe" systems as generally there were no reuse of software modules, of data formats, or of databases among projects. In other words, there was many software system but no ecosystem-wide platform. In addition, existing platforms were only used in a minority of the systems.

One explanation was the lack of an organizational structure in 2010 to govern a coordinated development effort between projects to ensure reuse and ensure a use of common formats and database infrastructures. The period was that of exploration and early experiments, often driven by local initiatives and funding.

Our study formed the basis for outlining architectural requirements for a framework/platform (Net4Care) that could serve as software structure for an ecosystem for telemedicine.

Specifically we identified three main issues, detailed in [16], in the current state that from a technical viewpoint inhibit a natural evolution of the ecosystem and that had to be addressed.

- *Lack of integration* among (tele-)medical systems. The systems are not integrated and often replicate data in diverse data formats and storage systems.
- *Missed opportunities for reuse*. All studied systems had essentially the same architecture, but all had built their own software modules, used proprietary data formats, and hosted own data centers.
- *Low buildability* of integrated telemedicine systems. Bass et al. [4] defines the architectural quality attribute buildability as: *Buildability . . . refers to the ease of constructing a desired system.*

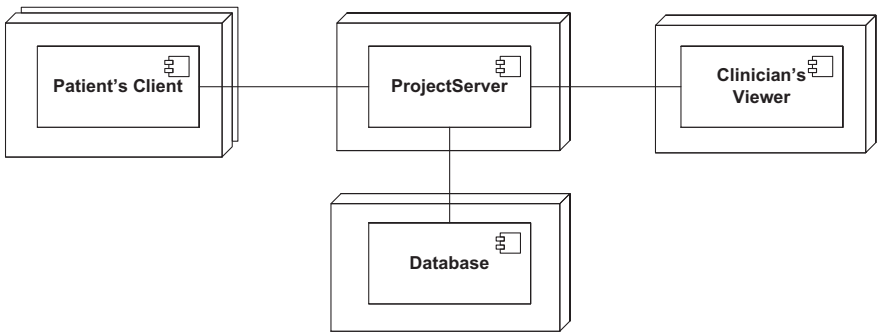


Fig. 5. Deployment view of typical telemedicine project.

The few systems that allowed integration used standards with very steep learning curves, there was a lack of tutorials and test environments, and it was closed source software, which made development prohibitory slow and expensive.

As an example of *lack of integration*, in a case of telemedical care of Implantable Cardio-Defibrillator (ICD) patients at Rigshospitalet in Denmark, doctors, and analysts had to consult up to 12 different applications at the hospital to cater for remote patients. Another example was a small SMB that had developed a system for *Chronic obstructive pulmonary disease (COPD)* patients that provides teleconferencing and digital measuring of spirometry and subsequent upload and review by the clinician. While successful from a clinical and patient point of view, the SMB hosted its own servers with the databases and the clinicians' system at the hospital had to be on the SMB's VPN. This lack of integration forced the clinician to copy and paste measured values manually into the hospital's EPR system. The SMB argument for this lack of integration with the regional hospital's infrastructure was mainly speed of deployment and maintenance and thus the overall perception of the company's solution.

Regarding *missed opportunities for reuse*, all studied systems has similar functionality (medical data upload, database storage and retrieval, browsing and review by clinicians, etc.) but all code modules were developed from scratch. The lack of a common, shared, infrastructure to reuse forced companies to build all components themselves. Thus, small to medium sized business with expertise in e.g. home care were essentially excluded for the market due to lack of resources to develop and host server side solutions.

As examples of *low buildability*, a PhD student in the Net4Care project spent more than 80 hours demonstrating that medicine data could be read for a telemedicine application from a national web service built on top of the National Service Platform (NSP) for healthcare integration [59]. Several sources contributed to the slow process. While documentation of the web service interface existed, it was not detailed enough to allow a valid SOAP message to be composed, the WSDL contained defects that made it impossible to generate client stubs, descriptive error codes from the service were missing making defect tracing cumbersome, and the security model code assumed non-existing certificates and provided yet another layer of required understanding. Moreover, no "server in a box" existed so learning and test programs depended upon test servers outside our control and whose characteristics were changed without notification making the developed software fragile.

The three issues are of course not orthogonal: increased reuse leads to faster and more reliable development and thus to improved buildability, and reusing storage systems and standardized data formats leads to easier integration. The issues, however, works strongly against a natural emergence and evolution of a software ecosystem. Without common and shared software architectural standards, data formats, storage systems, modules and information resources, development and integration is difficult, especially for small and medium sized business.

These issues must thus influence the software engineering practice and shape how a software ecosystem is created and evolved. The Net4Care ecosystem framework was specifically designed to address these issues, as detailed in Section 6.3 below.

5.4. The challenges that remain

To address issues in the telemedicine ecosystem, the government set up a task force in cooperation with the major actors among the ministries, the Danish Regions and Local Government Denmark ("Kommunernes Landsforening"). The aim was to develop an action plan for the deployment of telemedicine [23].

The plan outlines five national, large scale telemedicine projects, and in addition mentions the development of a national reference architecture for telemedicine [46]. The projects are intended to develop and deploy software, including some infrastructure elements and to provide experience with and evaluation of the software and organizational set-ups used. At the end of the experiments, the successful parts are to be deployed nationally.

The national plan is primarily intended to address two important issues: scale and quality. First of all, telemedicine solutions are now likely to be deployed on a national scale over the next five to ten years. Secondly, the quality of the solutions is likely to improve both with respect to software and to organization. This likely quality improvement is due to the fact that many actors will share the cost of developing and maintaining the software. Furthermore it will be easier for the actors to share experience on organization.

While the action plan tackles issues that need to be addressed, our analysis[4] points to some important concerns that should be addressed if the ecosystem is to grow and attract numerous players during the next years.

In short, the current ecosystem is suffering from weak organizational, business and software structures, and very weak links between them. This results in high transaction costs summarized by the following: ad hoc solutions to coordination and development (weak organizational structure and weak links), low attractiveness of markets (weak business structure not being addressed by the organizational structure) and low accessibility (weak software structure not being addressed by the organizational structure). Below we consider these in turn.

Ad hoc solutions: The current organizational structure is not able to effectively develop the software structure and the business structure as described in Sections 5.1 and 5.2. When major problems are identified ad hoc solutions, like the task force developing the action plan, are chosen and these solutions usually address problems in the software structure directly instead of developing the organizational structure to create effective links between the structures.

Low attractiveness of markets: Current markets for telemedicine in Denmark are fragmented and their characteristics are very difficult to identify and they are changing. Thus there are no credible long term plans for how to develop telemedicine at the regional level. The closest to such plans are the government plan for deployment of telemedicine and the Danish Regions Shared Indicators for the digitization of healthcare [18]. As described above the government plan mentions infrastructure elements and a five telemedicine projects under development and evaluation. The indicators of the regions list four of the five projects of the government plan. Thus, from a business as well as a hospital point of view, the current plans imply that the markets are more or less on hold until the experiments are finished and evaluated. At that point in time, if the projects are successful, a small number of telemedicine systems together with some infrastructure elements will be deployed nationally. If some of the experiments fail nothing is stated about what will happen. In neither case do the current plans provide a way forward for companies that want to market telemedicine systems in Denmark, except for the companies providing the systems used in the experiments.

The municipalities are several steps behind compared to the government and the regions due to the fact that up until now mainly hospital departments have driven the numerous telemedicine projects in Denmark. In a recent policy paper on telehealth [37], the Danish Municipalities outline a strategy and point to

[4] This analysis is done as part of the national project "Denmark as a telemedicine pioneer" and based on interviews and workshops with more than 50 companies.

some of the challenges facing the municipalities, including the need to supplement the "hospital/diagnosis/illness-specific" perspective with a "non-illness specific" perspective. In addition, the challenges include problems with existing economic models and the need for new ways to share costs between region and municipalities, poor equipment and system quality, and lack of system integration. In the long run the new focus on telemedicine and telehealth by the municipalities will almost certainly strengthen the development of the ecosystem. However, in the short term the most likely effect is that municipalities will do some more preparatory work before investing in telemedicine and -health. In summary the current business structure is weak and the ad hoc solutions at the level of the organizational structure do not create links that are capable of improving the development of the business structure.

Low accessibility: The companies that do decide to enter the market, and their customers, are faced with another challenge: the low accessibility of public healthcare infrastructure and services. This low accessibility is due to several factors: written documentation is sparse and not of high quality, often the rationale behind solutions is not obvious and no explanations available, and there is no support in terms of tutorials or "help desks". The result is that the learning curve for a company that wants to use the Health Data Network and/or some of the associated services like Shared Medicine Card[5] is very steep and often prevents SMBs from entering the market. In summary the current software structure is weak and the ad hoc solutions at the level of the organizational structure do not create links that are capable of improving the development of the software structure.

6. Danish telemedicine software ecosystem architecture design and realization

In the following section, we describe our experiment using the software ecosystem architecture concept introduced in Section 4.

During 2008, we came to realize the need for significant improvements of the Danish telemedicine ecosystem, c.f. Section 5.1. Our first attempts on setting up research and development projects to address the situation focused on two issues:

- Improving the software structure: we began to work on national infrastructure based on international standards and supporting integration between systems from different vendors, as well as increased international market potential.
- Improving the business structure: we tried to develop business models for different types of participants, e.g., SMBs with telemedicine products, providers of hardware, infrastructure companies e.g. telecoms, and the different providers of telemedicine services, including hospitals and municipalities.

As mentioned above and described below in more detail, our work on improving the software structure through national infrastructure based on international standards progressed well and we developed an effective set of tools and tutorials as the first elements of the technological platform of the ecosystem. We named the platform "Net4Care", after the project that provided most of the funding.

However, it turned out to be more difficult to create convincing results regarding improvement of the business structure, i.e. what would the benefit be for e.g. an SMB developing telemedicine products? A main challenge was that the links between the software structure, the organizational structure and the business structure were weak. Thus the impact of positive developments in software structure was slow to develop in the business structure. Concretely, paths from our research in terms of the Net4Care platform to uptake was quite long and uncertain since we had no direct connection to national or regional forums making decisions in the area, i.e. to the relevant parts of the emerging organizational structure.

In order to improve the situation and accelerate the development of the ecosystem, we decided to supplement our work on the software structure (work on the Net4Care infrastructure platform) and the business structure (work on business models), with an effort directed towards the organizational structure. Thus, we began to work on establishing a new open source foundation with the mandate to promote telemedicine and representing relevant, interested parties. The intention was that this new organization, 4S, should play a key role in the acceleration of ecosystem development as an orchestrator in the organizational structure.

6.1. Organizational structure

The new 4S organization should be an orchestrator responsible for the open source infrastructure platform for telemedicine, Net4Care (see Section 6.3) and have the qualities to maximize the likelihood of success. We made literature surveys on open source, healthcare, and software ecosystems [42] and studied involved Danish organizations and their plans, including public open source organizations. Based on this, we decided upon a set of characteristics of 4S. The characteristics covered both organizational structure aspects, such as the mission and structure of 4S and community building (including with healthcare professionals and patients), aspects related to business structure, e.g. developing viable business models, and aspects related to software structure, e.g. understandability and accessibility of software resources, and "traditional" open source organizational aspects, e.g., governance. During 2012, we then held a number of meetings and workshops with relevant, interested parties where we presented the ideas and the rationale behind 4S. The status after the initial round of discussions were:

Mission: There was general agreement on the need for an organization that could play the role of the orchestrator in the organizational structure and streamline the ecosystem around an infrastructure framework like Net4Care. The main critique concerned the likelihood of sufficient backing from major players and speed of uptake.

Structure of the organization: In order to speed up the formation of 4S and to create an agile, adaptable organization, we proposed a bottom-up approach where interested individuals from relevant organizations at the state, regional, and municipality levels agreed upon a proposal and then tried to secure organizational backing and membership. This in turn led to a structure where we could allow only a few organizations to be mandatory members from the start. On the other hand informal accept from most was important. There was general agreement that it would not be feasible to establish 4S through a process where all the major bodies at the state, regional, and municipality levels as well as business representatives were invited. There was some concern about the likelihood of convincing the needed mandatory members to participate.

Community building: Current Danish telemedicine initiatives are mainly controlled by IT managers and project leaders. This in turn often leads to dissatisfaction among the involved healthcare professionals and less than optimal results for the patients. To change this we proposed to create a number of healthcare communities as an important part of 4S. There was general agreement on—and only few discussions of—these issues. However, we ourselves became concerned about how to create this involvement, since the main elements in the technological platform of the ecosystem, i.e. the Net4Care framework, was about infrastructure. Thus, the platform

[5] https://fmk-online.dk.

itself did not seem to be a good vehicle for discussions of challenges in telemedical treatments of different types of illness.

Traditional open source organizational aspects: This concerned primarily how to create developer involvement and quality assurance. However, we also considered development plans/projects and their creation and prioritization to be very important. Especially we wanted to secure the influence of the different communities.

Understandability and accessibility: In our view, 4S had to provide a very high degree of understandability, accessibility, testability etc. of the elements of the technological platform in order to become a success. We already had some very good results in this area through the Net4Care website[6] and there was general agreement on this. Especially commercial companies could list numerous examples on non-understandability, non-accessibility and lack of test environments concerning existing healthcare infrastructure, cf. also Section 5.1 and Section 5.4.

Two important developments in the period from the end of 2012 until the late spring of 2013 changed the prospects for 4S in a very positive way. First, the Alexandra Institute and three other organizations were granted a large telemedicine contract by The National Board of Technology and Innovation with one of the authors as project leader. This provided resources for the work on establishing 4S, and – even more importantly – it gave new legitimacy to our efforts on making 4S an orchestrator in the Danish telemedical software ecosystem.

Secondly, a draft of the new national reference software architecture for telemedicine systems was published [46]. This reference architecture closely followed the architecture that we had worked with in Net4Care, and thus the platform was now viewed as an implementation of the reference architecture. The key elements of the platform (i.e., the Net4Care framework and test-environment) began to be used as a common technological platform in the software ecosystem: The framework and test environment was used as tools for both companies and healthcare service providers to evaluate and experiment with how different systems and components fit the reference architecture, how to modify them to increase compliance and how to design and implement new interface based on the standards of the reference architecture to increase data sharing.

These developments very directly impacted both the backing from major players, which were one of the outstanding issues, cf. "Mission" above, and the commitment from mandatory members, cf. "Structure of the organization" above. The only major outstanding issue was how to create viable communities.

The technological platform with the Net4Care framework and test environment provided a good basis for community building involving it people from both private companies supplying IT systems and from organizations delivering healthcare services, e.g. the regional IT departments hosting the IT systems of the hospitals in their region. However, it was not well-suited for organizing discussions around how to improve telemedicine for different groups of patients or citizens and for the healthcare professionals working with these groups. In other words, the platform, with its focus on the Net4Care framework and test-environment was not suited for this kind of community building. In order to advance the creation of active communities we decided to make software used directly by patients/citizens and healthcare professionals a primary concern of 4S.

One way to do this was to make agreements with a number of vendors supplying such systems and using the Net4Care framework and test-environment. We are working on this and expect it to be an area of activity that will grow steadily in the future. However, these telemedicine systems are closed source and setting up agreements with the vendors is a slow process. In addition most vendors considers experiments with and development of their systems as something that is company internal and under their tight control. In short, most vendors do not view their telemedicine systems as potential elements in a technological platform of a software ecosystem. And from a user perspective (patients/citizens and healthcare professionals), the path from a successful experiment to new versions of the commercial software to be improved via the experiments is very long and the conditions governing the company decisions are opaque.

Thus, we decided to pursue a different path as our main strategy: the inclusion of a major open source telemedicine platform as part of the technological platform for which 4S was responsible. We had for some time had discussions with the three regions responsible for those of the national large scale experiments that were developing infrastructure elements and a telemedicine system for handling data collection by the patients themselves, cf. Section 5.4. Backed by the alignment of the reference architecture with the Net4Care framework and test-environment as well as the legitimacy provided by the large telemedicine contract granted by The National Board of Technology and Innovation we managed to make an agreement with the three regions that the telemedicine platform in question, called Open Tele, would be handled by 4S. This agreement in turn made it attractive for the five groups of patients/healthcare professionals currently using systems based on the Open Tele platform to participate in communities organized by 4S.

6.2. Business structure

Fig. 6 shows the business model canvas for the 4S organization. The central value propositions are cheaper telemedicine IT development through ease of system integration and cross-sector and -supplier integration. This is to be achieved through platform development, (test) platform hosting, and ecosystem governance, the key resources include open source components (Net4Care and OpenTele) and integrated national healthcare services.

Fig. 7 shows the Telesår business model under the assumption that the Telesår project becomes part of the 4S ecosystem. This adds to the value propositions of the Telesår project in that IT development would arguably become cheaper (faster), result in a better integrated system, and be based on open standards. This should result in a changed cost structure in which establishment and running costs should be lowered (but a stakeholder fee to 4S should be added). As part of the Telesår project, there is now the potential that parts of the project can be made available as services/components to other telemedicine applications (see "Key Activities" and "Key Resources") with the possibility of use by others (see "Customer Segments" and "Revenue Streams").

6.3. Software structure

The 4S ecosystem's software structure elaborates on the Net4Care framework. This framework was architected based upon the three issues identified in current telemedicine: lack of integration, low buildability, and lack of reuse; and designed as best possible to fulfill the requirements of an *application-centric ecosystem* [6]. The other categories of ecosystems as defined by Bosch: operating system-centric and programming-centric; seemed less relevant for telemedicine. Bosch lists four essential success factors for an application-centric ecosystem:

1. The foremost success factor is a *large set of customers* or the promise of those customers.

[6] http://www.net4care.org.

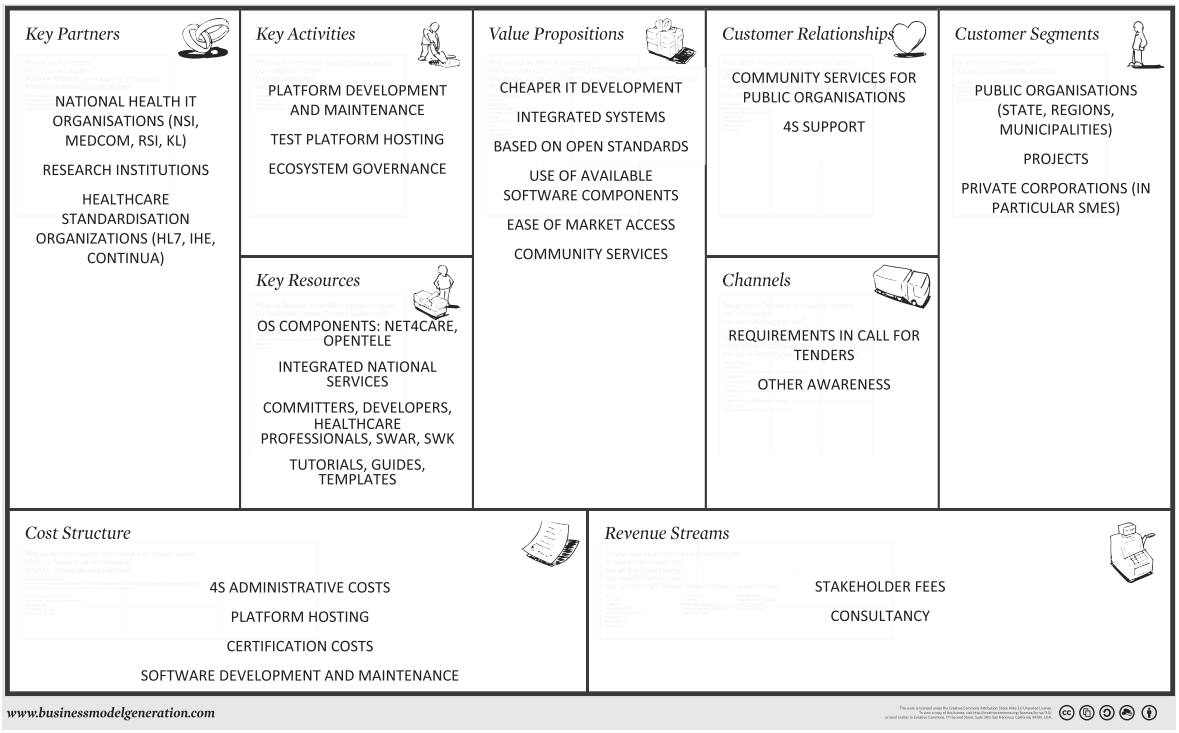


Fig. 6. 4S business model canvas.

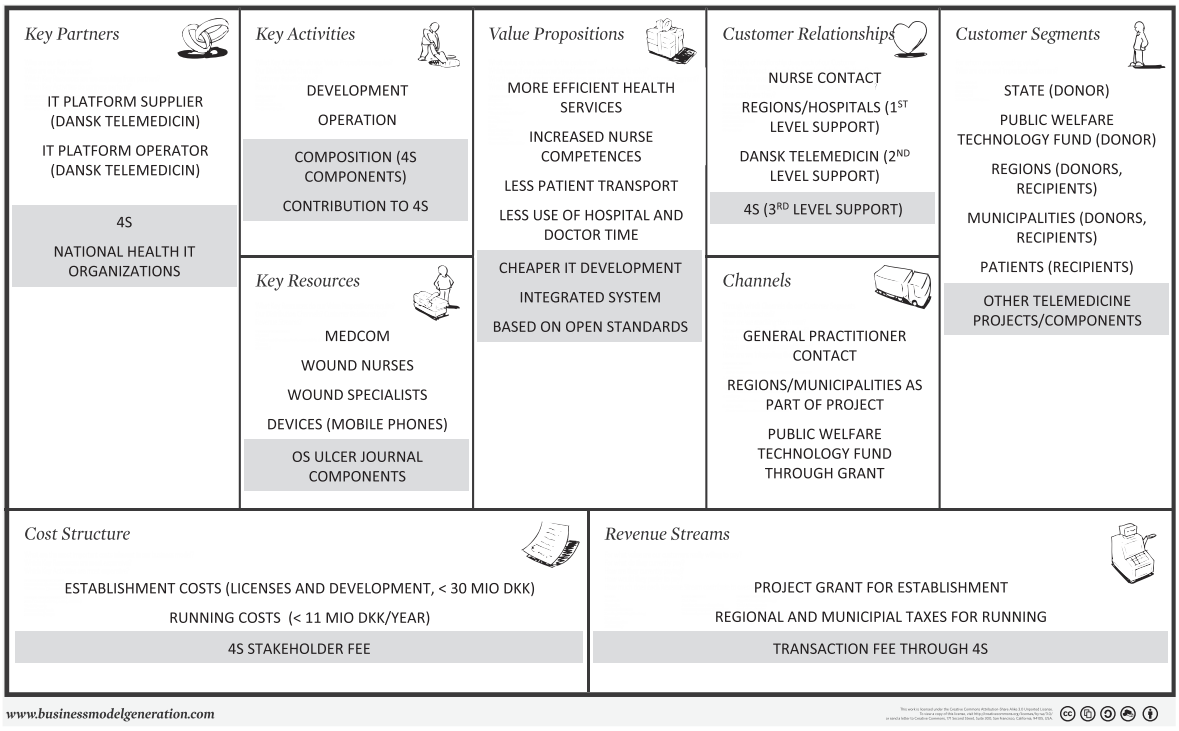


Fig. 7. Telesår and 4S business model canvas. Grayed blocks signify potential implications of 4S on the Telesår business model.

2. The platform company should *aim to simplify contribution by third party developers*, by popular development environments, stable and expressive interfaces, and easy deployment and integration.
3. The platform should provide solutions to *extend data models and work flows* as well as integrate into the same user experience.
4. The platform should provide a *viable sales channel* where third party contributions are exposed to customers.

Of these, points (2) and (3) are achievable by a strong commitment by the technical development team and realizable in the software structure, point (4) must be provided by the organizational

and business structure, and finally (1) can be hoped for from the quality of the efforts invested in all three structures.

Our analysis showed little focus had been on the simplification of contribution and extendability of data models in prevailing infrastructure and telemedicine pilot projects. Thus, our team set out to develop an architecture for the ecosystem platform with special attention to these aspects and inspired by the functional and architectural requirements elicited in the analysis phase.

The main architectural drivers became the following quality attribute scenarios [3]:

> *(QAS 1/Modifiability) An SMB developer with a strong background in electronics and hardware-near computing [source] wants to develop a telemedical application for the home that supports uploading measured clinical values [stimulus] using the Net4Care framework [artifact] as part of preliminary exploration and prototyping [environment]. The developer downloads, installs, and tests a first prototype having a full round-trip of clinical measurements (from device to simulated server and back again) [response] within four staff hours [response measure].*

This scenario captures an important aspect of Bosch's requirement of *simply contribution* namely the ability for third party to understand and develop using the framework.

> *(QAS 2/Integrability) A clinician [source] wants to review telemedical measurements of a patient [stimulus] using the regional Electronic Health Record (EHR) system [artifact] as part of daily work [environment]. The clinician can query, view, and annotate the data [response] without need of starting specialized telemedicine applications nor copy-pasting data between applications [response measure].*

This is another aspect of Bosch's second point, namely *easy deployment and integration*, in that the Net4Care platform seeks to avoid stove-pipe systems with its own clinician applications, databases and servers, and replace them with integration into existing clinical systems.

> *(QAS 3/Modifiability) An SMB developer [source] wants to support a new type of measurements in the home [stimulus] using the Net4Care framework [artifact] during development [environment]. The developer defines appropriate software modules with proper clinical encodings and integrates them into the Net4Care operational environment [response] within two weeks [response measure].*

This final architectural driving scenario captures the third Bosch requirement of *extend data models and work flows*. The scenario requires two parts. The first is that the technical framework contains adequate hotspots/variability points to allow customization and extending of data types, and the second is that there are certification processes in place that verify that the choices made concerning clinical encoding are clinically valid so the resulting data produced can integrate into EHR and other clinical systems.

6.3.1. Net4Care architecture and reference implementation

To support the main three architectural requirements in the form of the scenarios (as well as a set of architectural and functional requirements reported elsewhere [25]) we made the following central architectural decisions.

- *Information resources* primarily in the form of open source well proven and tested tutorials on the web site www.net4care.org provides a shallow learning curve for developers.
- *Reference implementation* as open source.
- *Staged testing environment* which provides a simple isolated test environment for fast development that seamlessly can be migrated into a full operations environment.
- *Clinical standards* used at the back tier for storage format (Continua Alliance *Personal Health Monitoring Record* (PHMR) [52] which is a HL7 "Clinical Document Architecture" standard [5]) as well as for database system (XDS.b *Cross-Enterprise Document Sharing* [66] which is a standard for clinical storage systems).

An overview of the functional run-time view of the Net4Care framework is shown in Fig. 8. We follow Bass et al. [3][§9]'s recommended practice to represent the component-connector view using UML object notation: Boxes represent run-time components, single line boxes are (passive) objects while boxes with an extra vertical line represents processes (active objects). Links represent connectors that mediate data and control flow at run-time between components. Verbs on the links describe the type or role of data/control exchanged.

Note that while object notation in its UML interpretation only represent one of potentially many different instances of a system, Fig. 8 represents the general configuration of run-time elements that all instances of Net4Care will have. For instance, any HomeClientApplication will have any number of StandardTeleObservations but only one DataUploader, etc.

In the figure, components marked by light gray are those the SMB developer must develop, while the rest are provided by the Net4Care framework. The web site's information resources provide detailed tutorials regarding download, installation, and first exposure ("Hello World") in using and configuring the framework. The home client framework is available in Java and C#, while the server is purely Java based.

The *SMB Comp* is our abstraction over all code that is not related to the Net4Care framework except for the fact that it must produce medical measurements, like blood pressure, weight, spirometry measurements, etc. Such measurements we denote *telemedical observations*. Thus it is in this component that the SMB will invest time and effort to have a competitive edge in the market for telemedicine: ease of use, appealing interface, intuitive device coupling, etc. Though it is represented by a single gray box, it should by far be the greatest investment by the SMB as Net4Care should ideally handle the rest.

The *ObservationSpecifics* is the framework's main variability point with regards to the concrete types of telemedical observations that the SMB wants to support. This requires some insight into clinical informatics, notably code systems which are used to uniquely identify the type of measurements made. Here, we have invested special attention to provide guides, tutorials, and code support that allows non-expert users of Net4Care to produce clinical valid data while shielding them as best possible from the full details of HL7 which has a very steep learning curve.

The *DataUploader* is a standard client side class that handles all the transport to the application and storage tiers. The Net4Care server is responsible for translating the core data from the client side into full and validated PHMR documents and store them in national/regional XDS.b. Once stored, clinician systems like EHR can retrieve data using standard XDS.b profiles.

The *ServerConnector* protocol between the client system and the Net4Care server and the *XDS.b Adapter* component in the server are also architectural variability points and serve the requirement of a staged testing environment. These can be configured by the SMB developer in stages, where the simplest uses local, simplified, variants which allows fast and non-distributed development. Progressively more complex variants are provided by the Net4Care platform, like an HTTP connection to a OSGi-based webserver, and ultimately the production ready variants using secure HTTPS connections and real XDS.b interfaces. These variability points thus support both QAS 1 and 3.

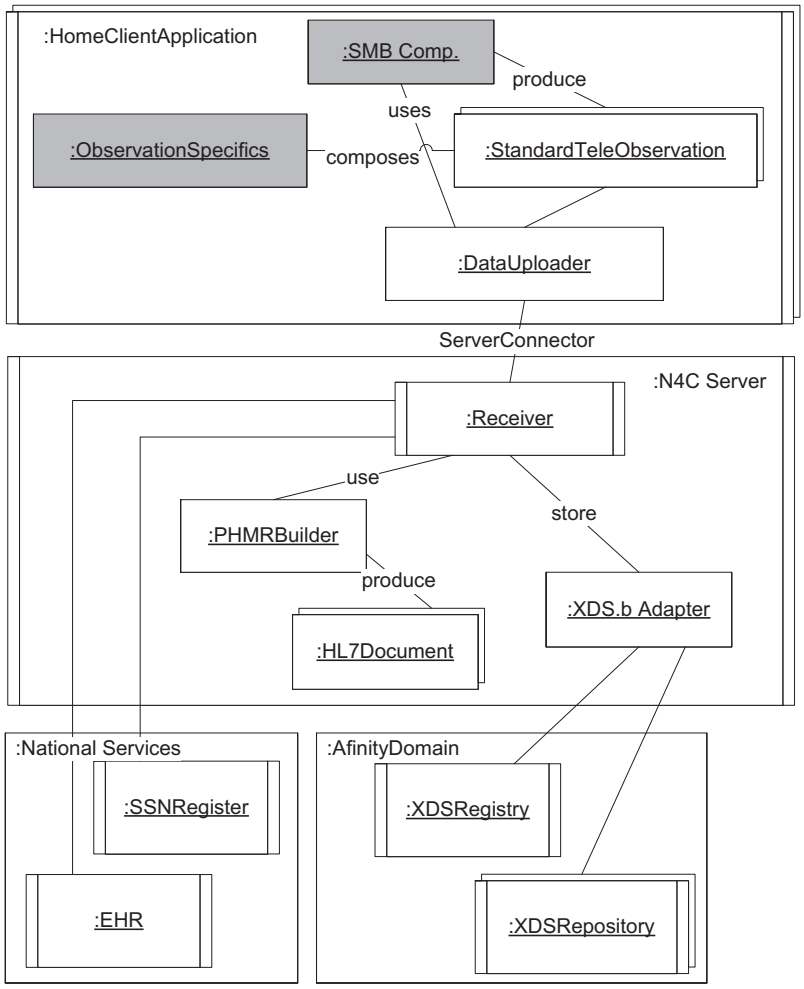


Fig. 8. Functional view of Net4Care. UML object notation is used to represent the run-time structure of any instance of the framework. The notation used is explained in detail in the text.

One lesson learned from the development of the software structure is that the engineering practice for software ecosystems must be aligned with lessons learned from analysis of successful application-centric ecosystems. Notably, the requirements of *simplified contribution by 3rd party* and *extensible data models and work flows* can be supported at the software framework level if included early in the architectural analysis. Moreover, these points must have strong focus in the creation and evolution of the ecosystem.

6.3.2. The Open Tele data collection telemedicine system

As described above, we decided to include an open source data collection system called Open Tele in addition to Net4Care. The idea was to use Open Tele to generate interest among healthcare professionals and citizens/patients and thus supplement the interest among IT people generated by Net4Care. Basically, Open Tele corresponds to the box "SMB Comp" in Fig. 8. The Open Tele system includes a tablet used by citizens/patients for data collection, including device that can be coupled to the tablet. In addition, the system provides local data storage and access for healthcare professionals.

The system is a new addition to the ecosystem and to 4S, and we do not yet have much experience with it. However, we expect it to play two important roles. First of all it will allow us to set up experiments together with healthcare professionals and citizens/patients where we are free to choose which parts of the system we want to work with. And, if successful, the results of an experiment may be included in the Open Tele software system if the necessary resources for implementation according to the 4S quality requirements can be provided. In addition, commercial vendors working with closed source are free to use the same results to improve their systems. Secondly, we plan to further develop Open Tele to work together with an app store approach to open a new channel for contributions.

In terms of the essential factors from Bosch, Open Tele contributes significantly to a large set of customers or the promise of those customers primarily through its enabling of community building. The app store approach will both simplify contribution by third party developers and provide a viable sales channel. Finally the possibilities of Open Tele in combination with Net4Care provides solutions to extend data models and work flows as described in Section 6.3.1 above.

7. Evaluation

The 4S software ecosystem is now being realized. Obviously the quality of the Net4Care framework and its compliance with the national reference architecture is a key element in this. But our discussions with actors and other stakeholders indicates that the establishment of 4S as a credible orchestrator has been, and is, crucial to the evolution of the ecosystem.

Our primary evaluation to date, however, has been through the use of the Net4Care framework. This framework has successfully

fulfilled the aspect of *"technological platform . . .that allows involvement and contribution of the different actors"* and *"with symbiotic relationships"* [42]. Table 3 shortly outlines involved actors and their main interests, which we will expand upon below.

The actors so far can coarsely be classified in three classes: private companies, regional hospitals and their healthcare IT departments, and educational institutions, each having their own interests and perspectives.

Sekoia [56] is a small company (with less than 20 employees) that develops IT solutions for healthcare with strong commitment to user-driven innovation to ensure high quality in use of their products. With this emphasis they are less inclined for large investments into back tier development such as hosting data warehouses and integration with existing clinical systems. Thus the Net4Care framework is well suited to handle back tier aspects and presently integration experiments with their Android tablet end-user solutions is explored.

The interest from the regional healthcare IT departments is mainly in the experiences gained and software modules developed to handle PHMR and XDS.b. One key aspect at present is augmenting measured values in the home with context information, for instance if a measurement is uploaded then who actually made the measurement: was it the patient alone, aided by a home nurse, or perhaps under the supervision of a trained clinician? This issue (which is irrelevant in a hospital setting) demonstrates how the symbiotic relationships between actors are important, as the issue was first raised by clinicians collaborating with Sekoia and thus brought to the attention of the architectural team for Net4Care and onwards to the larger group of actors. Thus the primary interest of the regional healthcare is learning and interaction.

Finally, the framework has been used by educational institutions, primarily for student master-level projects. The engineering school's projects had device integration as primary focus area and the framework thus supplied the back tier at a very modest learning cost. We have interviewed several of the groups and they agree that our QAS 1 (on page 25) is indeed fulfilled as getting the first "Hello World" application running is easy and the tutorials provide sufficient detail for getting their prototype work going without too much hassle. Also for their (simple) use, they found the framework's support for handling new types of measurements adequate, QAS 3. However, for adding context data as required by Sekoia, QAS 3 was not fully covered.

8. Discussion and future work

4S is now acting as an orchestrator and the Net4Care framework exhibits features of an ecosystem technical platform, especially with respect to the symbiotic relationships and interaction of the actors. However, we still await contributions in terms of software modules that can be used across the ecosystem. At present, software is built "on top" of Net4Care for the individual actor and largely for learning, testing, and experimentation, more than for contribution.

Additional studies of the applicability of the software ecosystem architecture concept to different ecosystems are required. The concept itself is not specific to telemedicine, and we would expect the types of analyzes and designs we have made to generalize to other types of ecosystems, i.e., that our study has external validity. We expect our work to be most useful in areas where there is no obvious orchestrator and/or no obvious candidate for doing the type of integrated analysis and design that the software ecosystem architecture approach supports. This view is supported by the fact that numerous sets of publicly funded open source software exists in Denmark, but their role and use is very limited, and no orchestrator exists. This is similar to the situation described in Section 5 and we find it likely that an analysis and subsequent design initiatives based on our approach will have a reasonable chance to improve the situation.

Reliability, i.e., the extent to which our study can be repeated is another probable issue. Since our case study is revelatory and our experiment depends on many parameters that are outside of our control (e.g., the involved actors and their behavior), it is questionable that it can be reproduced with the same results. However, as discussed above in relation to external validity, we find it likely that the software ecosystem architecture approach may be used successfully in ways similar to the work presented in this paper, but in different areas. This could be done e.g. in situations where funding is available for developing an open source platform, but no obvious orchestrator and/or no well-functioning organizational, business and/or software structure exist. In the following section we present a set of guidelines for people interested in this type of use.

8.1. Analysis and design guidelines

We provide a set of general guidelines on how a similar study can be carried out. We separate our guidelines in two sections: *analysis*, concerning the modeling of a software ecosystem, and *Design*, concerning the steps towards designing a software ecosystem.

8.1.1. Analysis

Phase A1: Identify and characterize the software ecosystem using the elements of the theory: technological platform, actors, and software build on the platform. Identify and characterize elements and relations of the three structures: organizational, business, and software. Select models to be used in describing them.

Questions to consider include:

- Are some expected types of actors, elements and/or relations missing, under- or overrepresented?
- Where in its evolution is the ecosystem (e.g., emerging, new, or old)?

Phase A2: Analyze and understand the software ecosystem: Collect and analyze information on the elements and relations of the three structures. Focus on the sustainability of each structure,

Table 3
List for actors in Net4Care.

Actor	Type	Size	Interest/contribution
Sekoia, Silverbullet	Companies	Small	Explore as integration system
Trifork/next step citizen	Company	Small	Use module; explore as integration system
CSC; Systematic	Companies	Large	Explore as integration system
CGI	Company	Large	XDS.b learning and integration
Region Nord; region Midt; region Hovedstaden	Hospitals	Large	PHMR and XDS.b learning
Aarhus University (School of Engineering); University of Southern Denmark	Universities	Large	Use as back-end
Aarhus University; University of Copenhagen	Universities	Large	Ecosystem/architectural case

and interactions among the structures, especially concerning issues.

Questions to consider include:

- Is the software ecosystem stable, unstable, or evolving?
- How are benefits distributed among the actors?
- What are the entry conditions and incentives to join?

Phase A3: Evaluate the results of analysis. Present and discuss results of analysis with actors and other stakeholders. Possibly repeat A1 and A2.

8.1.2. Design

Below we list the phases of designing an emergent ecosystem. The design is using the output of the analysis of the existing ecosystem. The steps are inspired by the steps for carrying out an experiment described by Wohlin et al. [63].

Phase D1: Scoping of design: Identify and characterize the goals of the design. Using the ecosystem analysis as input, focus on the areas of the ecosystem under development that appeared to be missing or problematic and define what purposes the design is going to serve. These purposes could relate to sustainability of the structures, e.g., overcoming issues identified through analysis.

Phase D2: Create a design. Develop a design consisting of one or more new or revised elements, e.g. software platform, orchestrator, keystone, or distribution channels. Develop a process plan for realizing the design. Since evolving an ecosystem is outside the control of any one actor, contingency planning should be considered.

Questions to consider include:

- Are the new/revised elements likely to create a process that will make the ecosystem meet the goals?
- How robust is the design, i.e. will all parts of the design have to realized?
- Who are the key actors whose participation is crucial for realization?

Phase D3: Evaluate and revise the design.

Discuss goals and design with key actors and other stakeholder, i.e., those whose support is crucial to the realization of the design.

Questions to consider include:

- Do actors agree with the goals?
- Do actors believe that realization of the design is feasible?
- Who will participate?
- Will the necessary resources be available?

Phase D4: Realize the design. Execute the plan and monitor the evolution of the ecosystem.

Phase D5: Evaluate the ecosystem. Model the software ecosystem after the design realization.

Questions to consider include:

- Have the identified areas improved?
- Has the general ecosystem health sustainability improved?
- Is the ecosystem providing more opportunities to the actors than before?
- Are there other areas that need improvement?

8.2. Future work

The recent addition of the Open Tele data collection system adds new groups of actors in addition to those listed in Table 3. The primary interest of the new groups of healthcare professionals and citizens/patients is to explore possibilities for improving and extending the system. Thus the first concrete activity will be a small series of evaluation and design workshops to provide input to further development of the Open Tele system.

Similarly, seminars are being planned for companies interested in the app store approach. In addition to software aspects of apps we have begun to look at healthcare and economic aspects. In one end of the spectrum is a category of apps where only traditional evaluations have been done, e.g., of stability and security issues. Somewhere in the middle are apps where clinical evaluations of effects have been carried out (and is made available through the app store). Finally we are considering a category of apps where citizens can be referred to receive them for free. These aspects are primarily related to the three administrative levels described in Section 5.1. 4S already has good relations to both the state and the regional levels and the relations to the level of municipalities are under development.

In addition to these new groups, several of the existing actors in Net4Care are also interested in Open Tele. Companies delivering systems to coordinate interaction among healthcare organizations providing service to patients having a chronic condition are considering Open Tele as a way to add data collection capabilities. However, for some of the companies making telemedicine systems Open Tele is a competitor. We expect that 4S will be able to provide a sufficient active and creative environment to make the pros more important than the cons. In any case we will have to be very explicit about the different roles of Net4Care and Open Tele in the ecosystem.

Finally, a number of municipalities are planning to acquire versions of Open Tele with a range of smaller additions/modifications. Currently the expectation is that the new/modified software will become part of the open source software managed by 4S.

9. Conclusion

The first contribution of this article has been to define the concept of 'software ecosystem architecture' as "the set of structures needed to reason about the software ecosystem, which comprise actor and software elements, relations among them, and properties of both"; identify the organizational, business, and software structures of software ecosystems as central; and apply these concept to the analysis of the current Danish telemedicine ecosystem, identifying challenges and opportunities in this.

The second contribution of the article has been to, within the framework of software ecosystem architecture, to present the design, realization, and (partial) evaluation of a software ecosystem, 4S, for telemedicine. The development of healthcare IT systems in general, and telemedicine applications in particular, is hindered by (i) complex problem and solution domains, by (ii) limited economies of scale, and by (iii) problems of integration and interoperability.

The 4S software ecosystem aims at (i) reducing the complexity of problem and solution domains by providing reusable components and services that incorporate national and international standards; at (ii) improving economies of scale by providing a platform (Net4Care and OpenTele) on which to build telemedical applications; and at (iii) increasing integration and interoperability by providing components to access national services in an interoperable way.

Finally, the Net4Care technological platform has showed how an emphasis on key success factors for application-centric software ecosystems, that of *simplified contribution* and *extensible data models and work flows*, can be supported by adopting a software engineering practice focusing on well-written on-line information resources, a staged testing environment, and an open source reference implementation, and thus serve as the software structure

basis for gaining interest, commitment, and symbiotic relationships among actors.

Acknowledgements

The research reported in this article has been supported by the Net4Care project that is funded by Caretech Innovation,[7] the Connect2Care project that is funded by the UNIK Partnership,[8] and the project "Kick-start of Denmark as telemedicine pioneer country" that is funded by The National Board of Technology and Innovation.[9]

References

[1] O. Barbosa, C. Alves, A systematic mapping study on software ecosystems, in: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, 2011, pp. 15–26.
[2] O. Barbosa, R.P. Santos, C. Alves, C. Werner, S. Jansen, Software ecosystems – analyzing and managing business networks in the software industry, in: Jansen et al. [29], Chapter: A Systematic Mapping Study on Software Ecosystems from a Three-Dimensional Perspective, 2013, pp. 59–81.
[3] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, second ed., Addison-Wesley, Boston, MA, USA, 2003.
[4] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, third ed., Addison-Wesley, Boston, MA, USA, 2013.
[5] K.W. Boone, The CDA Book, Springer, 2011.
[6] J. Bosch, From software product lines to software ecosystems, in: Proceedings of the 13th International Software Product Line Conference (SPLC '09), Carnegie Mellon University, Pittsburgh,PA, USA, 2009, pp. 111–119. <http://portal.acm.org/citation.cfm?id=1753235.1753251>.
[7] J. Bosch, Architecture challenges for software ecosystems, in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), ACM, New York, NY, USA, 2010, pp. 93–95. http://dx.doi.org/10.1145/1842752.1842776.
[8] J. Bosch, Architecture in the age of compositionality, in: M. Babar, I. Gorton (Eds.), Software Architecture, Lecture Notes in Computer Science, vol. 6285, Springer, Berlin/Heidelberg, 2010, pp. 1–4. doi:10.1007/978-3-642-15114-9_1.
[9] J. Bosch, P. Bosch-Sijtsema, Coordination between global agile teams: from process to architecture, in: Agility Across Time and Space, Springer, Berlin, Heidelberg, 2010, pp. 217–233. doi:10.1007/978-3-642-12442-6_15.
[10] J. Bosch, P. Bosch-Sijtsema, From integration to composition: on the impact of software product lines, global development and ecosystems, J. Syst. Softw. 83 (1) (2010) 67–76. SI: Top Scholars. <http://www.sciencedirect.com/science/article/B6V0N-4WPJ5XY-1/2/e69f658f21dfa50b9d1aa468a6cfb46d>.
[11] J. Bosch, P.M. Bosch-Sijtsema, Softwares product lines global development and ecosystems: collaboration in software engineering, in: Collaborative Software Engineering, Springer, Berlin, Heidelberg, 2010, pp. 77–92. doi:10.1007/978-3-642-10294-3_4.
[12] V. Boucharas, S. Jansen, S. Brinkkemper, Formalizing software ecosystem modeling, in: Proceedings of the 1st International Workshop on Open Component Ecosystems (IWOCE '09), ACM, New York, NY, USA, 2009, pp. 41–50. http://dx.doi.org/10.1145/1595800.1595807.
[13] C. Burkard, T. Widjaja, P. Buxmann, Software ecosystems, Business Inform. Syst. Eng. 4 (2012) 41–44. http://dx.doi.org/10.1007/s12599-011-0199-8.
[14] P.R.J. Campbell, F. Ahmed, A three-dimensional view of software ecosystems, in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), ACM, New York, NY, USA, 2010, pp. 81–84. http://dx.doi.org/10.1145/1842752.1842774.
[15] H.B. Christensen, M. Christensen, K.M. Hansen, M. Kyng, K. Manikas, M. Surrow, S. Urazimbetova, Requirements for a software-intensive ecosystem for telemedicine, in: Med@Tel 2012: Global Telemedicine and eHealth Updates, vol. 5, 2012, pp. 423–427.
[16] H.B. Christensen, K.M. Hansen, Net4care: towards a mission-critical software ecosystem, in: WICSA/ECSA, IEEE, 2012, pp. 224–228.
[17] J. Clemensen, S.B. Larsen, N. Ejskjaer, Telemedical treatment at home of diabetic foot ulcers, J. Telemed. Telecare 11 (suppl 2) (2005) 14–16.
[18] Danske Regioner, Regionernes fælles pejlemærker for digitalisering af sundhedsvæsenet. fra strategi til handling: Nye pejlemærker i perioden fra 2014 til 2016, 2013, in Danish.
[19] E. den Hartigh, M. Tol, W. Visscher, The health measurement of a business ecosystem, in: Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting, 2006, pp. 1–39.
[20] N. Denzin, The Research Act: A Theoretical Introduction to Sociological Methods, McGraw-Hill, 1978.
[21] EgenJournal, CITH Co-constructing IT and Healthcare, 2010. <http://www.cith.dk>.
[22] FMK, Fælles medicinkort, 2010. <http://www.ssi.dk/Sundhedsdataogit/National%20Sundheds-it/Faelles%20Medicinkort.aspx>.
[23] Fonden for Velfærdsteknologi, National handlingsplan for udbredelse af telemedicin, June 2012, in Danish.
[24] K. Hansen, M. Ingstrup, M. Kyng, J. Olsen, Towards a software ecosystem of healthcare services, in: Proceedings of the 3rd International Workshop on Infrastructures for Healthcare: Global Healthcare, 2011. pp. 28–31.
[25] K.M. Hansen, The Net4Care platform – version 0.3, Tech. rep., Net4Care, 2012.
[26] HL7, Hl7: health level seven international, 1987. <http://www.hl7.org>.
[27] M. Iansiti, R. Levien, The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy∗ Innovation∗ and Sustainability, Harvard Business Press, 2004.
[28] ISO 42010, Systems and software engineering – architecture description, ISO/IEC/IEEE 42010:2011(E), 2011.
[29] S. Jansen, S. Brinkkemper, M. Cusumano (Eds.), Software Ecosystems – Analyzing and Managing Business Networks in the Software Industry, Edward Elgar, Cheltenham, UK, 2013.
[30] S. Jansen, S. Brinkkemper, A. Finkelstein, Business network management as a survival strategy: a tale of two software ecosystems, in: First International Workshop on Software Ecosystems (IWSECO-2009), Citeseer, 2009, pp. 34–48.
[31] S. Jansen, S. Brinkkemper, A. Finkelstein, Software ecosystems – analyzing and managing business networks in the software industry, in: Jansen et al. [29], Chapter: Business Network Management as Survival Strategy, 2013, pp. 29–42.
[32] S. Jansen, S. Brinkkemper, J. Souer, L. Luinenburg, Shades of gray: opening up a software producing organization with the open software enterprise model, J. Syst. Softw. 85 (7) (2012) 1495–1510. <http://www.sciencedirect.com/science/article/pii/S0164121211003013>.
[33] S. Jansen, A. Finkelstein, S. Brinkkemper, A sense of community: a research agenda for software ecosystems, in: 31st International Conference on Software Engineering – Companion Volume, 2009 (ICSE-Companion 2009), May 2009, pp. 187–190.
[34] R. Kazman, M. Gagliardi, W. Wood, Scaling up software architecture analysis, J. Syst. Softw. 85 (7) (2012) 1511–1519. Software Ecosystems. <http://www.sciencedirect.com/science/article/pii/S0164121211000793>.
[35] T. Kilamo, I. Hammouda, T. Mikkonen, T. Aaltonen, From proprietary to open source-growing an open source ecosystem, J. Syst. Softw. 85 (7) (2012) 1467–1478. Software Ecosystems. <http://www.sciencedirect.com/science/article/pii/S0164121211001683>.
[36] T. Kilamo, I. Hammouda, T. Mikkonen, T. Aaltonen, Software ecosystems – analyzing and managing business networks in the software industry, in: Jansen et al. [29], Chapter: Open Source Ecosystem: A Tale of Two Cases, 2013, pp. 276–306.
[37] Kommunernes Landsforening, Kommunernes strategi for telesundhed, April 2013, in Danish.
[38] P.B. Kruchten, The 4 + 1 view model of architecture, IEEE Softw. 12 (6) (1995) 42–50.
[39] M. Lungu, M. Lanza, T. Gîrba, R. Robbes, The small project observatory: visualizing software ecosystems, Sci. Comput. Programm. 75 (4) (2010) 264–275. Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008). <http://www.sciencedirect.com/science/article/B6V17-4X6MSPV-3/2200/91f82400b828c4fe2aa6bc04551d0a57>.
[40] K. Manikas, K.M. Hansen, Characterizing the Danish telemedicine ecosystem: making sense of actor relationships, in: Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems, ACM, 2013, pp. 211–218.
[41] K. Manikas, K.M. Hansen, Reviewing the health of software ecosystems–a conceptual framework proposal, in: Fifth International Workshop on Software Ecosystems (IWSECO-2013), CEUR-WS, 2013, pp. 33–44.
[42] K. Manikas, K.M. Hansen, Software ecosystems – a systematic literature review, J. Syst. Softw. 86 (5) (2013) 1294–1306.
[43] J.D. McGregor, A method for analyzing software product line ecosystems, in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), ACM, New York, NY, USA, 2010, pp. 73–80. http://dx.doi.org/10.1145/1842752.1842773.
[44] MedCom, Overview of danish telemedical initiatives, 2013. <https://medcom.medware.dk/telemedicine_projects> (accessed December 2013).
[45] D. Messerschmitt, C. Szyperski, Software Ecosystem: Understanding an Indispensable Technology and Industry, MIT Press Books 1, 2003.
[46] National Sundheds-It, Referencearkitektur for opsamling af helbredsdata hos borgeren, Version 0.18 in public hearing, April 2013, in Danish.
[47] Net4Care, Net4care website, 2014. <http://www.net4care.org> (accessed April 2014).
[48] M. Olejaz, A.J. Nielsen, A. Rudkjøbing, H.O. Birk, A. Krasnik, C. Hernández-Quevedo, Denmark: health system review, Health Syst. Trans. 14 (2) (2012) 1–192.
[49] A. Osterwalder, The Business Model Ontology: A Proposition in a Design Science Approach, Ph.D. thesis, University of Lausanne, 2004.
[50] A. Osterwalder, Y. Pigneur, Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers, Wiley, 2010.
[51] M.Q. Patton, Qualitative Research and Evaluation Methods, SAGE Publications, Inc., 2002.
[52] PHMR, Implementation Guide for CDA Release 2.0 Personal Healthcare Monitoring Report (PHMR) (International Realm) Draft Standard for Trial Use Release 1.1, October 2010.

[7] http://www.caretechinnovation.dk.
[8] http://www.partnerskabetunik.dk.
[9] http://www.alexandra.dk/dk/projekter/sider/kickstart-af-danmark-som-telemedicinsk-foregangsland.aspx.

[53] R. Robbes, M. Lungu, A study of ripple effects in software ecosystems (nier track), in: Proceedings of the 33rd International Conference on Software Engineering (ICSE '11), ACM, New York, NY, USA, 2011, pp. 904–907. http://dx.doi.org/10.1145/1985793.1985940.
[54] RRS, Remote rehabilitation support, 2009. <http://www.caretechinnovation.dk/en/projects/rrs.htm>.
[55] R.P. Santos, C.M.L. Werner, A proposal for software ecosystem engineering, in: Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS, 2011, pp. 40–51.
[56] Sekoia, Sekoia website, 2014. <http://www.sekoia.dk> (accessed April 2014).
[57] TELEKAT, TELEKAT: Telehomecare, chronic patients and the integrated healthcare system, 2007. <http://www.telekat.eu>.
[58] Telesår, Telemedicinsk sårvurdering, 2006. <http://www.pleje.net>.
[59] S. Urazimbetova, Experiences of integration of telemedicine to national services, Tech. rep., Net4Care, 2012.
[60] J. van Angeren, V. Blijleven, S. Jansen, Relationship intimacy in software ecosystems: a survey of the dutch software industry, in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, MEDES '11, ACM, New York, NY, USA, 2011, pp. 68–75. http://dx.doi.org/10.1145/2077489.2077502.
[61] I. vanden Berk, S. Jansen, L. Luinenburg, Software ecosystems: a software ecosystem strategy assessment model, in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ECSA '10, ACM, New York, NY, USA, 2010, pp. 127–134. http://dx.doi.org/10.1145/1842752.1842781.
[62] ViewCare, Viewcare, 2011. <http://www.viewcare.com>.
[63] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer, 2012.
[64] B. Womack, Google says 700,000 applications available for android, October 2012. <http://www.businessweek.com/news/2012-10-29/google-says-700-000-applications-available-for-android-devices> (accessed April 2014).
[65] World Health Organization, Telemedicine. opportunities and developments in member states, Report on the second global survey on eHealth, 2010.
[66] XDS.b, IT Infrastructure Technical Framework, Volume 1 (ITI TF-1), Integration Profiles, Revision 9.0, August 2012.
[67] R.K. Yin, Case Study Research: Design and Methods, fifth ed., SAGE, 2013.