Casper Lassenius
Kari Smolander (Eds.)

LNBIP 182

Software Business

Towards Continuous Value Delivery

5th International Conference, ICSOB 2014
Paphos, Cyprus, June 16–18, 2014
Proceedings

Springer

Lecture Notes
in Business Information Processing 182

Casper Lassenius
Kari Smolander (Eds.)

Software Business

Towards Continuous Value Delivery

5th International Conference, ICSOB 2014
Paphos, Cyprus, June 16-18, 2014
Proceedings

Springer

Volume Editors

Casper Lassenius
Aalto University, Finland
E-mail: casper.lassenius@aalto.fi

Kari Smolander
Lappeenranta University of Technology, Finland
E-mail: kari.smolander@lut.fi

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-319-08737-5 e-ISBN 978-3-319-08738-2
DOI 10.1007/978-3-319-08738-2
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014942261

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Software business refers to commercial activities in and around the software industry, aimed at generating income from the delivery of software products and software services. Although the software business shares common features with other international knowledge-intensive businesses, it carries many inherent features making it a challenging domain for research. In particular, software companies have to depend on one another to deliver a unique value proposition to their customers or a unique experience to their users. Moreover, recent developments like the emerging app economy offer a variety of opportunities for entrepreneurs and/or start-up companies. The recent acquisition of the three-year–old Finnish mobile game start-up Supercell with the total value of 2.2 billion euro shows that the future of software is not only in utility, productivity, connection, and interchange, and but also in entertainment and free time of people globally. This will have a profound effect on software business and requires novel business models and new approaches to software product development as well. A new paradigm that approaches simulation and education/training via games is also attracting attention and becoming economically visible.

This volume contains the papers presented at ICSOB 2014: the 5th International Conference on Software Business held during June 16-18, 2014, in Paphos, Cyprus. To acknowledge the constantly changing landscape of software business, which requires flexibility and continuous business changes, we selected as the conference theme "Shortening the Time-to-Market—From Short Cycle Times to Continuous Value Delivery," reflecting the contemporary trend toward ever shortening deployment cycles in particular for service-based software business. ICSOB 2014 addressed researchers and practitioners who are concerned with software business in different ways, as well as the start-up community, which is increasingly focusing on mobile and social software. ICSOB is a series of annual conferences born in 2010. The previous conferences were held in Boston (USA), Brussels (Belgium), Jyväskylä (Finland), and Potsdam (Germany).

This year's two keynotes spanned both the reach and the new developments in the software business economy:

"Is There Anything That Isn't Software?", by Mike Hinchey, Director, Lero—the Irish Software Engineering Research Centre

"What Other Industries Could Learn from the Games Industry", by KooPee Hiltunen, Director, Neogames Finland Asociation

The conference received 45 submissions. Each submission was reviewed by at least two, typically three, Program Committee members. Of the 45 submissions, the committee decided to accept 18 full, two short, two industrial, and two doctoral consortium papers. For full papers, this gives an acceptance rate of 42%. The accepted papers follow diverse methodologies, and represent the diversity in research in our community.

The papers span a wide range of issues related to contemporary software business—from strategic aspects and ecosystems to software development and business models. We have arranged the program into eight sessions that together provide a good insight into current software business research. The industry papers are included at the end of the proceedings.

We would like to extend our warm thank you to the members of the Program Committee, who did a fantastic job in reviewing the papers, ensuring the quality of the conference, as well to the local organization team, whose engagement was essential in making this event a special experience. Furthermore, we extend our warm thank you to the local organizers led by Prof. George Papadopoulos for inviting us to beautiful Cyprus.

We sincerely hope that you enjoy the conference and have productive discussions!

May 2014

Tiziana Margaria
Casper Lassenius
Kari Smolander

Organization

Program Committee

Sergey Avdoshin	Higher School of Economics, Moscow, Russia
Jan Bosch	Chalmers University of Technology, Sweden
Sjaak Brinkkemper	Utrecht University, The Netherlands
Peter Buxmann	Technical University of Darmstadt, Germany
Michael Cusumano	MIT, USA
Torgeir Dingsøyr	SINTEF, Norway
Brian Fitzgerald	University of Limerick, Ireland
Samuel Fricker	Blekinge Institute of Technology, Sweden
Peter Gloor	MIT, USA
Georg Herzwurm	University of Stuttgart, Germany
Thomas Hess	University of Munich, Germany
Slinger Jansen	Utrecht University, The Netherlands
Philippe Kruchten	University of British Columbia, Canada
Thomas Kude	University of Mannheim, Germany
Olli Kuivalainen	Lappeenranta University of Technology, Finland
Casper Lassenius	Aalto University, Finland
Ulrike Lechner	Universität der Bundeswehr München, Germany
Ricardo J. Machado	Universidade do Minho, Portugal
Andrey Maglyas	Lappeenranta University of Technology, Finland
Tiziana Margaria	University of Potsdam, Germany
Rory OConnor	Lero - The Irish Software Engineering Research Centre, Ireland
Samuli Pekkola	Tampere University of Technology, Finland
Wolfram Pietsch	Aachen University of Applied Sciences, Germany
Maryam Razavian	VU University Amsterdam, The Netherlands
Bjorn Regnell	Lund University, Sweden
Dirk Riehle	University of Erlangen-Nürnberg, Germany
Matti Rossi	Aalto University, Finland
Guenther Ruhe	University of Calgary, Canada
Stefan Seidel	University of Liechtenstein
Kari Smolander	Lappeenranta University of Technology, Finland
Klaus-Dieter Thoben	University of Bremen, Germany

Juha-Pekka Tolvanen	MetaCase
Pasi Tyrväinen	University of Jyväskylä, Finland
Krzysztof Wnuk	Lund University, Sweden

Additional Reviewers

Alves, Carina
Bacelar Pinto, Eduardo
Dalpiaz, Fabiano
Eling, Nicole
Förderer, Jens
Harnisch, Stefan
Heumüller, Erich
Ho, Jason
Holmström Olsson, Helena
Hubert, Simon
Kabbedijk, Jaap
Kowalczyk, Martin
Lima, Ana
Nykänen, Jussi
Radzuweit, Martin
Roshan, Maryam
Salgado, Carlos
Santos, Nuno
Shahnewaz, Shawn
Spruit, Marco
Stol, Klaas-Jan
Van Angeren, Joey
Vlaanderen, Kevin

Doctoral Consortium Abstracts

Innovation Initiatives of Large Software Companies

Henry Edison
Free University of Bozen,
Piazza Università 1, Bolzano 39100, Italy
henry.edison@stud-inf.unibz.it

1 Problem and Research Questions

Most large companies are struggling to seek a way to continuously innovate in the dynamic and threatening environment. The strategy experts have switched from traditional innovation to value-creation economy [1–3]. Over the decades, corporate entrepreneurship (CE) has been proposed as the silver bullet to large organisation stagnancy and lack of innovation. However, Ross [4] found that due to its complexity and structure, modern and large organisations limit innovation and changes occurring within themselves. When this happens, their market position will be overtaken by new small innovative firms. Thus, our research questions are:

RQ How can innovation initiatives be executed in large software companies?
RQ1 What are the innovation initiatives introduced in large software companies?
RQ2 How do those initiatives structure and execute the main activities?
RQ3 How can lean startup principles be adopted to improve the the existing innovation initiatives in large software companies?

2 Related Work

Study by Sharma and Chrisman [5] found that the common feature of CE is the creation of new business within business. However, little is known in implementing CE activity in the context of large software companies. Organisations need to find and keep good entrepreneurs internally to identify the opportunities and change them to successful business, just like a start up. Eric Ries [2] introduced the concept of lean startup as a new way of entrepreneurship in disruptive innovation. We only find one industry report of applying lean startup in software context [6]. There is no comprehensive process design of adopting lean startup in large software companies. A research on this direction could help industry practitioners to introduce better lean startup in their organisations.

3 Methods

A mixed methods research is employed to address the research questions. To answer RQ1 and RQ2, systematic mapping study (SMS)[7] will be performed

to capture big picture of innovation initiatives in large software companies from literature. To validate this result, quantitative survey to industry practitioners will be conducted. For RQ3, we plan to perform multiple case studies to collect empirical data about innovation initiatives as well as to validate the result of RQ1 & RQ2.

4 Preliminary Results and Next Steps

As the preliminary result, based on the key person or team involved in, the initiative might take place inside the companies or outside the companies. Inside the companies, we found two types of innovation initiatives of large software companies: free and organised entrepreneurship. In free entrepreneurship, the initiative is introduced by the employee or intrapreneur if she gets support from management, otherwise it becomes bootlegging or underground work. In organised entrepreneurship, management takes the responsibility to setup the initiative either through expert system e.g. R&D or empowerment e.g. ICV (internal corporate venture), acquisition, internal project, subsidiary. The initiatives occurred outside the companies are spin-off, crowdsourcing and capital venturing.

5 Next Steps

We are preparing the report of SMS studies and we will publish it. In the mean time, we are also preparing the instrument to do qualitative survey and case studies. We also start looking for and contacting the potential companies to perform case studies.

References

1. Cooper, B., Vlaskovits, P., Ries, E.: The lean entrepreneur: how visionaries create products, innovate with new ventures, and disrupt markets. John Wiley and Sons (2013)
2. Ries, E.: The lean startup: how today's entrepreneurs use continuous innovation to create radically successful business. Crown Business (2011)
3. Kuratko, D.F., Hornsby, J.S., Covin, J.G.: Dignosing a firm's internal environment for corporate entrepreneurship. Business Horizons 57, 37–47 (2014)
4. Ross, J.: Corporations and entrepreneurs: paradox and opportunity. Business Horizons 30(4), 76–80 (1987)
5. Sharma, P., Chrisman, J.J.: Toward a reconciliation of the definitional issues in the field of corporate entrepreneurship. Entrepreneurship theory and practice 23(3), 11–27 (1999)
6. May, B.: Applying lean startup: an experience report. In: Proceedings of Agile Conference, pp. 141–147 (2012)
7. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proceedings of 12th International Conference on Evaluation and Assessment in Software Engineering, vol. 17, pp. 1–10 (2007)

Early-Stage Software Startups: Linking Business Strategies to Software Development

Carmine Giardino

Free University of Bolzano Piazza Domenicani, Bolzano, 39100, Italy
camine.giardino@unibz.it

1 Problem and Research Question

Software startups are newly created company, with no operating history and oriented in producing cutting-edge technologies. These companies develop software under highly uncertain conditions, tackling fast growing markets with severe lack of resources [1]. Despite their increasing importance in economy, there are only few scientific studies attempting to address software engineering (SE) issues, especially in their early-stage development activities (i.e. from idea conception to the first open beta release) [2]. With our research we aim to understand how business and project-level strategies can effectively be aligned focusing on those activities which better deliver customer value proposition, from a SE perspective.

2 Related Work

Evidences show how software development, especially in early-stage startups, is at the core of the company's daily activities [1, 3], most oriented towards Lean and Agile methodology [3]. As resources are limited, understanding the relationships between the business value and project-level strategies is a prerequisite for putting the development effort in the right direction [4]. Nonetheless, little attention has been given on how development activities might effectively be aligned to business goals over-time [5].

3 Methods

Evolutionary in nature and able to address a wider spectrum of topics, we are currently adopting a grounded theory approach [6], which comprises a systematic mapping study, surveys, case studies, and an online project which collects startups information about business and development executions (available at www.widcy.com). Verification of the data will be conducted by triangulation of the collected data and a systematic comparison to the state-of-art - related work established in small and medium software companies.

4 Preliminary Results

The first systematic exploration of the state-of-art on software startup exhibits a weak body of knowledge of 43 studies poor in their rigor and relevance [2]. On

that basis, we investigated thirteen startups, discovering a primary concern towards faster time-to-market in contempt of an increasing technical debt [7]. However, speeding-up time-to-market not necessarily bring achievements business-wise. Two case studies discovered how prematurely launching the product to market can hinder learning practices (e.g. continuous customer feedback) despite they could provide better insights of unpredictable market conditions and fast-changing technological challenges [3].

5 Next Steps

Starting from the idea conceptualization, we aim to discover how a entrepreneurial vision (i.e motivations and emotions to start up a company) can affect development activities, and how clarifying it can align the achievement of business goals to project-level strategies in early-stage software startup companies.

References

1. Sutton, S.M.: The role of process in software start-up. IEEE Software 17(4), 33–39 (2000)
2. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: A systematic mapping study. Information and Software Technology (2014)
3. Coleman, G., O'Connor, R.: An investigation into software development process formation in software start-ups. Journal of Enterprise Information Management 21(6), 633–648 (2008)
4. Blank, S.: Why the Lean Start-Up Changes Everything. Harvard Business Review 91(5), 64 (2013)
5. Giardino, C., Wang, X., Abrahamsson, P.: Why early-stage software startups fail: A behavioral framework. In: Lassenius, C., Smolander, K. (eds.) ICSOB 2014. LNBIP, vol. 182, pp. 27–41. Springer, Heidelberg (2014)
6. Corbin, J., Strauss, A.: Grounded theory research: Procedures, canons, and evaluative criteria. Qualitative Sociology 13(1), 3–21 (1990)
7. Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: The Greenfield Startup Model. IEEE Transactions on Software Engineering (forthcoming)

Table of Contents

Strategic Aspects

Startups and Software Business

Products and Service Business

Software Development

Ecosystems

Platforms and Enterprises

Industry Session

Short Papers

Exploring the Relationship between Partnership Model Participation and Interfirm Network Structure: An Analysis of the Office365 Ecosystem

Joey van Angeren[1], Slinger Jansen[2], and Sjaak Brinkkemper[2]

[1] Department of Industrial Engineering and Innovation Sciences,
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
J.v.Angeren@tue.nl
[2] Department of Information and Computing Sciences, Utrecht University
Princetonplein 5, 3508 TB Utrecht, the Netherlands
Slinger.Jansen,S.Brinkkemper@uu.nl

Abstract. Platform owners face complex decisions in managing an ecosystem of third-party application developers. Little is known about the effect platform governance has on the productivity, or degree of interaction, among complementors in the ecosystem. The presented study of the Microsoft Office365 ecosystem investigates the extent to which participation in the partnership model of Microsoft influences productivity and embeddedness of complementors by means of network analysis and statistical inference. Results show the Office365 ecosystem is populated by 550 complementors that developed 1204 applications and initiated 787 interfirm relationships. Statistical inference reveals that increased productivity and participation in the Microsoft Certified Partner Network coincide with increased embeddedness, implying retention of complementors results in more cohesive ecosystems. Yet, partnership model participation does not result in increasing productivity of complementors. Results presented in this paper provide increased understanding of the influence of partnership models on ecosystem structure, to the benefit of practitioners and academia.

Keywords: app store, ecosystem governance, industry platform, lock-in, network analysis, software ecosystem, software platform.

1 Introduction

The number of industry platforms and app stores in the software industry is rising, with the Apple App Store and Google Play as most imaginative examples. An industry platform is defined by Gawer [1] as *"a product, service or technology that is developed by one or several firms, that serves as a foundation upon which other firms can build complementary products, services or technologies."* Apart from mobile platforms, such enterprise platforms as SAP BusinessOne, Google

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 1–15, 2014.

Apps and Salesforce.com also started to thrive on third-party application development. In platform-based markets indirect network effects are prevalent, meaning that an increase in available complementary applications will increase the installed base of the platform and vice versa [2]. Accordingly, the platform owner becomes dependent on an ecosystem populated by the developers of software products and the interfirm relationships among them [1,3].

Platform owners have a plethora of instruments at their disposal to govern their platform and the ecosystem around it. In studying these instruments, prior research predominantly focused on their outcomes in market settings [4,5,6], whereas the impact on the ecosystem remained unexplored. Addressing this deficiency, this paper examines the network structure of the Microsoft Office365 ecosystem by means of an exploratory study. It explores the influence of commonly applied [1,4] active retention or locking of complementors on the number of interfirm relationships among complementors in the ecosystem. The study regards increasing development activity of a complementor or its status of Microsoft partner as invoked by fostering complementor lock-ins, and assesses the influence thereof on the number of initiated interfirm relationships. As such, this paper answers the following research question: *"What is the influence of complementor lock-ins on the network structure of a proprietary platform ecosystem?"*

This paper presents a visualization and analysis of the ecosystem around the business productivity platform Office365 by means of network analysis [7,8] and statistical inference. The study builds on our previous work [9] that explored the interfirm network topology of the Google Apps ecosystem. It makes use of data collected from the Office365 app store (Office365 Marketplace), CrunchBase and websites of complementors. Through the analysis of the Office365 ecosystem, this research gains insight into the factors that shape proprietary platform ecosystems (i.e. a platform that is closed source and owned by a single for-profit entity). In addition, this research adds to the growing body of research on industry platforms and the multitude of means for governance that exists.

The remainder of this paper continues with an overview of related literature and network metrics in Section 2. An elaboration upon the research approach is included in Section 3. Section 4 presents a description of the Office365 ecosystem and its population. Section 5 analyzes the Office365 ecosystem and among others evidences that complementor lock-ins are positively related to the network density of the ecosystem, while the development scope of complementors remains narrow and unaffected. A discussion of validity is presented in Section 6, followed by a summary of findings and suggestions for future research in Section 7.

2 Background

The amount of literature on industry platforms, software ecosystems and network analysis is extensive, yet application of their theories in synergy is limited. This section first presents a brief review of contemporary literature on industry platforms and ecosystem governance. Thereafter, network metrics are identified and their computation is formalized. Herein, parallels with their application in existing literature are drawn whenever possible.

Platform owners face complex and far reaching decisions in the management of their platform and its surrounding software ecosystem. Jansen, Brinkkemper and Finkelstein [3] define a software ecosystem as *"a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them."* In the management of an ecosystem, according to West [10] platform owners face the challenge of reaping benefits from a proprietary technology, while the success of this technology is determined by the extent to which complementors and end-users adopt the platform. While most prior work has focused on fostering 'coherent' persistent third-party development and stimulating a continuous influx of new complementors [11,5,3], recent advances came to challenge the ever cumulative advantage of indirect network effects. Boudreau [12] observed that excessive numbers of new entrants hampered innovation in the handheld gaming industry, while Zhu and Iansiti [6] found that the influence of indirect network effects is often overestimated.

These progressing insights put stronger emphasis on retention of existing complementors, not provoking unnecessary competition, or in more general terms preserving the health of the ecosystem. The health of an ecosystem is determined by the capability of an ecosystem to; persistently produce meaningful outputs (productivity), survive market disruptions (robustness) and create niches in the ecosystem (niche creation) [13,14]. Elaborated approaches to managing an ecosystem include enforcing platform exclusivity for applications [11], managing diversity of available applications [12] and facilitating interaction among members of the ecosystem [4]. One of the proposed instruments to govern the ecosystem are partnership models [15,16]. To participate in a partnership model, complementors pay annual partnership fees and adhere to mandatory product and resource certification. In return, partners receive benefits that may include marketing benefits, participation in joint partner events and greater benefit from niche creation. Based on case studies conducted at SAP, Open Design Alliance and Eclipse Foundation, Van Angeren, Kabbedijk, Popp and Jansen [16] posit that stimulating active participation in the partnership model is associated with increased productivity and engagement of complementors in the ecosystem. Even more so because complementors are stimulated through the provision of incentives to further commit, specialize and integrate [17,11,4,3,9].

Graph theoretical mathematics developed a plethora of means to address the quantitative inquiry of structural properties of interfirm networks [7,8]. Several measures such as network density, centrality and clustering coefficients are useful in the analysis of ecosystem structure. The remainder of this subsection defines and formalizes these measurements as well as it illustrates their application in ecosystem analysis, for the purpose of replication and uniformity.

Network density reflects the ratio of the number of interfirm relationships that are present in an ecosystem compared to the number of relationships that can theoretically be initiated [7,8]. As such, it is an indicator of the degree of interconnectivity and collaboration among the inhabitants of an ecosystem [18]. Densely interwoven interfirm networks have been labeled as being more robust [13] or specialized [17]. Network density is calculated as

$$\Delta = \frac{2E}{V(V-1)} \tag{1}$$

where E is the number of interfirm relationships in the ecosystem and V represents the number of complementors. The measure can take values between *0* (empty graph) and *1* (fully connected graph).

Degree centrality is a measure to evaluate the structural position of complementors in an ecosystem. The structural position of a complementor is important, as a large number of interfirm relationships eases access to complementary knowledge and fosters exchange or innovation [19]. Degree centrality is denoted as the ratio of the number of relationships a complementor has to the number of relationships it could theoretically have [20]. The measure is computed as follows

$$C_i = \frac{\sum_{j=1}^{n} a_{ij}}{E_i - 1} \tag{2}$$

where a_{ij} denotes a relationship between complementor i and complementor j, which is valued **1** if present and **0** otherwise. In the operationalization of business ecosystem health measurement, Den Hartigh, Tol and Visscher [14] characterize partners with a high degree centrality as 'healthy' compared to their peripheral peers.

Centralization is an indicator for the extent to which an ecosystem is centralized around one or more members, and as such it reflects the ratio of how central the most central member of the network is compared to the centrality of all other members [20,8]. Platform ecosystems will display high centralization, due to the strong dependence of complementors on the platform owner [9]. However, fluctuations in the centralization score may indicate the presence of other catalysts in the ecosystem. Centralization is computed as follows, where C_{max} is the degree centrality of the platform owner.

$$C = \frac{\sum_{i=1}^{n} C_{max} - C_i}{max\left(\sum_{i=1}^{n} C_{max} - C_i\right)} \tag{3}$$

Clustering coefficient reflects the extent to which the connections of a complementor are also connected to one another [21]. The clustering coefficient of the entire ecosystem is the average of all values across the network. Clustering coefficient indicates the degree to which an ecosystem can be divided into clusters of complementors that are tightly connected. the computation of the clustering coefficient of a complementor is performed as follows

$$CC_i = \frac{2e_i}{K_i(K_i - 1)} \tag{4}$$

where e_i is the number of interfirm relationships among connections of a complementor, and K_i is the number of connections of complementor i. Strongly

clustered networks have been labeled as innovative [19] or healthy [14]. Meanwhile, Iyer, Lee and Venkatraman [17] argue that specialized ecosystems will reflect a greater degree of clustering. The explanation for the greater degree of clustering is twofold: first specialization increases the need for interfirm collaboration [17] and second the initiation of interfirm relationships is based on direct complementarity and therefore more applicable to specialized settings [18].

3 Research Approach

In this research, the Office365 complementors, their characteristics and the interfirm relationships among them were subject of an exploratory study. Office365 is a business productivity suite that bundles independently modifiable versions of Microsoft Office, Microsoft Lync, Microsoft Exchange and SharePoint for use by enterprises and governmental or educational institutions. Third-party applications are found on a dedicated part of the bigger Microsoft Pinpoint Marketplace[1], and can be listed either globally or in one of the 59 region-specific marketplaces. The app store holds metadata for each application, of which *application identifier*, *application name*, *application category*, *release date*, *developer name* and *developer website* were deemed relevant for this research. Metadata of applications listed under the category *"Applications"* was collected, thereby excluding professional services and on-premises extensions from the dataset.

By means of a web crawler, all application specific metadata was extracted from the app store over sixty iterations (one to retrieve the globally listed applications, and 59 to traverse all region-specific marketplaces). The web crawler combined a set of scripts in Java programming language, and was derived from a crawler developed by Burkard, Widjaja and Buxmann [22]. It has been previously used in our exploratory study of the Google Apps ecosystem [9]. The metadata was collected in two stages: (1) an initial identification and (2) the actual collection of the metadata. During the initial identification, application identifiers were collected by systematically traversing all application categories up to the point at which no new applications were found. Then, all application specific metadata could be read-in from their publicly accessible information pages. All metadata was saved in a central MySQL database by means of predefined pattern templates.

After the removal of duplicates from the database, a list of complementors was compiled by means of an SQL query. Contrary to existing studies on interfirm relationships where proprietary alliance databases are used as central data source [17,19,18], the interfirm relationships were obtained directly from the websites of complementors by archiving mentions of partnerships. Archived interfirm relationships included *alliances*, *technological partnerships*, *collaborative research and development*, *strategic partnerships* and *partnership model participation*. To provide for triangulation of evidence, additional interfirm relationships were obtained from the openly accessible company database CrunchBase[2]. This

[1] `http://office365.pinpoint.microsoft.com`
[2] `http://www.crunchbase.com`

approach was preferred to move beyond alliances alone, which are likely to reflect industry-wide collaborations. Interfirm relationships were treated as binary and symmetric ties, maintained in an adjacency matrix[3]. Furthermore, complementor websites were used to assess whether it was currently participating in the partnership model offered by Microsoft (Microsoft Certified Partner Network).

In the course of analysis, network graphs were drawn to visualize the ecosystem. Its structural properties were computed in accordance with the network metrics that were formalized in the preceding section. Apart from network analysis [8], subsequent analysis was performed by means of bivariate statistics that include independent samples T-tests and Pearson correlations. The measurement of complementor productivity was operationalized as the number of applications a complementor developed, whereas complementor embeddedness was measured as the number of interfirm relationships initiated by a complementor.

4 Descriptives of the Office365 Ecosystem

This section provides a description of a snapshot of the Office365 ecosystem that was constructed on *13-02-2013*. The remainder of this section first elaborates upon the complementors and their characteristics, followed by a description of the Office365 ecosystem.

4.1 Complementors

The Office365 Marketplaces contained *1204* applications developed by *550* complementors (*278 (50.50%)* of which are participating in the Microsoft Certified Partner Network). For an application to be included in the Office365 Marketplace it has to be subjected to technical compatibility and complementary value requirements, with which Microsoft reserves the right to refuse inclusion of applications not of direct added-value to the platform. On average, each complementor develops *2.18* applications with a standard deviation of *1.65*, reflecting a narrow development scope per complementor. Applications include business templates for Microsoft PowerPoint, document management functionality for Microsoft Outlook, integrations with third-party software and customer relationship management or human resource functionality extensions. Noteworthy is that Microsoft itself is not involved in the development of complementary applications, as opposed to platform owners such as Google [9] or Intel [23] studied in prior work. The largest complementor in the ecosystem is *Net2xs* that lists *39* applications, followed by *Bamboo Solutions* and *Orlandoś VBA and Excel Site* that developed *32* applications. Meanwhile, *67%* of complementors develop one application. A complete distribution of these figures is included in Figure 1. Important to note is that the distribution of complementors may be influenced by

[3] An adjacency matrix is a square matrix with ecosystem inhabitants as rows as columns. Entries in the adjacency matrix, denoted as a_{ij}, indicate the inhabitants that are interrelated (i.e. adjacent). In a symmetric adjacency matrix, the value of a_{ij} is equal to a_{ji}.

Microsoft listing each extension for a Microsoft Office component as a separate application. When a complementor, for example, develops an application that works with Microsoft Word, Excel and PowerPoint, it can be included three times in the Office365 Marketplace. Consequently, the actual number of applications for Office365 may be lower than the number recorded in this research.

Table 1. Distribution of Office365 complementors based on the number of applications developed

# of applications	# of complementors
39	1
32	2
23	1
22	1
16	1
14	1
13	1
11	3
10	3
9	5
8	4
7	7
6	13
5	10
4	23
3	24
2	82
1	368

The entry date of a complementor was derived from the release dates of its applications, and is considered equal to *the release date of its first application*. Because the Office365 Marketplace does not provide a release date for all applications, to maintain validity a date of entry for a complementor could only be determined if the release date for *all its applications* was known. In total, a date of entry was recorded for *350 (63.64%)* complementors. At the time of measurement, on average *two* years and *seven* months have passed since their entrance into the ecosystem, with a standard deviation of *eight* months. The first recorded entry into the ecosystem dates back to 2005.

Every complementor is assumed to have initiated an interfirm relationship with Microsoft, since they extend the Office365 platform and include their applications in the app store. In total, the ecosystem is connected by *787* interfirm relationships. Every complementor on average initiated *1.43* relationships with a standard deviation of *11.74* relationships. Discarding the interfirm relationships initiated with the platform owner, complementors are connected through *0.43* relationships per complementor (standard deviation of *1.37* interfirm relationships). The most well

embedded complementors are *Dell (eleven applications)* with *37* relationships and *Nintex (seven applications)* with *32* connections.

4.2 Ecosystem

The Office365 ecosystem consists of complementors (nodes) and the interfirm relationships among them (edges). The structural properties of the ecosystem are summarized in Table 2. As expected, the Office365 ecosystem is centralized and sparsely connected with a network density of *0.5%*. These values, however, reflect slight differences compared the network density of *0.25%* and centralization of *99.95%* that were recorded for *993* complementors in previous work for Google Apps [9]. Despite the limited interconnectivity among inhabitants of the ecosystem the overall clustering coefficient is high, indicating that most complementors that do initiate interfirm relationships intensively interact.

Table 2. Network level descriptives for the Office365 ecosystem

Metric	Value
Size	551
Network density	0.00500
Centralization	0.9984
Modularity	0.336
Clustering coefficient	0.773

Table 3 includes node level properties of the Office365 ecosystem. The high standard deviation for degree centrality reveals the presence of a small number of embedded complementors in the ecosystem. Yet, *70.18%* of complementors are solely connected to Microsoft and found in the periphery. This finding may be partly explained by the presence of established SharePoint complementors in the Office365 ecosystem, the on-premises version of which already came with an extension architecture. This may well have positively influenced the number of initiated interfirm relationships for SharePoint complementors.

Table 3. Network metrics for the Office365 ecosystem

Metric	Min.	Max.	Avg.	Std. dev.
Degree centrality	0.00183	1	0.00519	0.0427
Clustering coefficient	0.00215	1	0.773	0.228

A useful network visualization could be created by moving beyond the hub-and-spoke network topology. Accordingly, the dataset was 'cleansed' by removing Microsoft, and the complementors solely connected to Microsoft. Figure 1 shows the resulting network, in which node sizes are proportional to the number of

applications developed per complementor. The grouping and shades represent clusters in the ecosystem, identified by means of the modularity algorithm [24]. The algorithm treats clusters as groups of complementors that are densely connected to one another while sparely linked to the rest of the ecosystem.

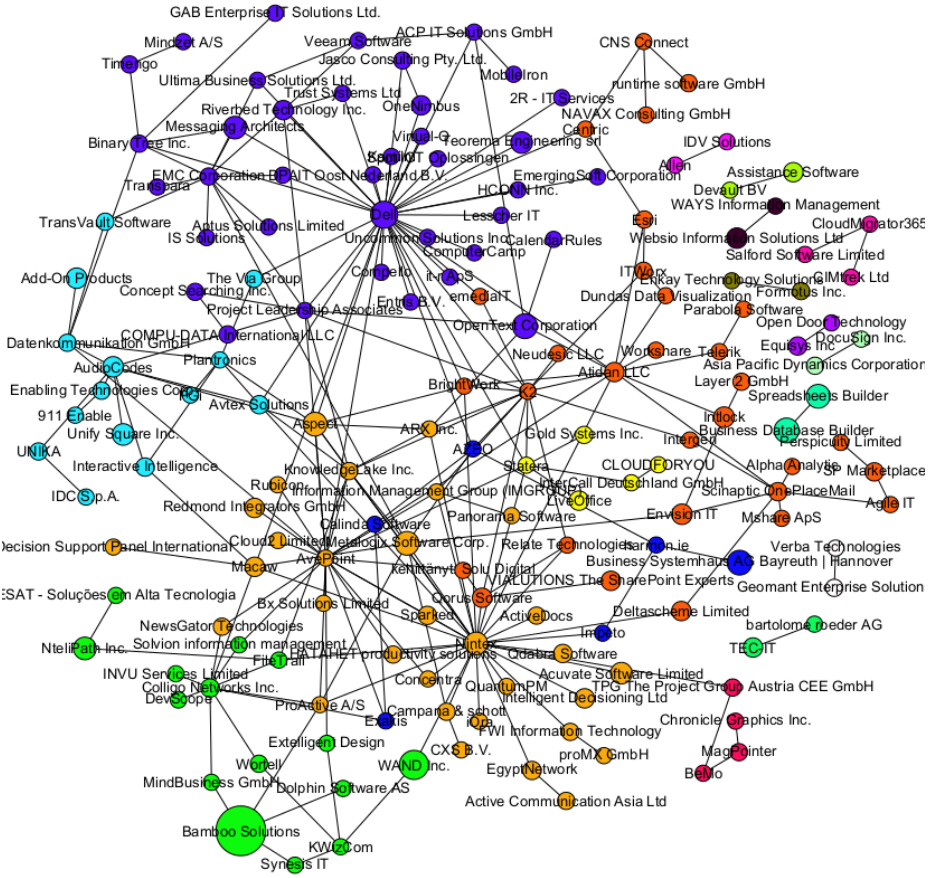

Fig. 1. Network visualization of clusters in the Office365 ecosystem

Figure 1 visualizes the interfirm relationships among *164 (29.82%)* complementors. Noteworthy is that many of the most productive complementors are present in the network visualization. Of the thirty most productive complementors, thirteen are present in Figure 1. Closer inspection reveals that absent complementors are individuals rather than enterprises. Individual developer *Orlando's VBA and Excel Site*, for instance listed *32* applications in the Office365 app store. The clusters in the ecosystem appear to be well and densely interwoven, apart from the dyads shown in the right of Figure 1. Interfirm relationships

seem to pertain technological partnerships, as most clusters appear to lack a geographical focus.

5 Analysis

This section presents an analysis of the Office365 ecosystem. It inducts propositions derived from the empirical data described in the preceding section by means of bivariate statistical inference. The analysis aims to gain a deeper understanding of the relationship between partnership model participation and the productivity and embeddedness (i.e. the number of initiated interfirm relationships) of complementors. The remainder of this section first regards the causal relationship between the productivity of a complementor and the interfirm relationships it initiates, followed by an assessment of the influence of partnership model participation on complementor productivity and embeddedness.

5.1 Productivity and Embeddedness

In the operationalization of ecosystem health measurement [13,14] and platform governance alike [11,9], the increasing productivity of a complementor has been hypothesized to coincide with greater embeddedness. As the complementor starts developing more applications, it will search for interfirm relationships such as technological partnerships, shared research and development or alliances to ease its access to resources or foster its influential position in the ecosystem [8,17,11]. Meanwhile, new entrants seek technological integrations with existing applications to increase their visibility in the market or to establish chains of interoperability [13,17]. Platform owners benefit from this increased network density as it fosters relational lock-ins (i.e. embedded complementors are less likely to depart the ecosystem) [14], increases consensus among complementors [11], and benefits the stability of the ecosystem [13].

While a causal relationship between complementor productivity and the number of interfirm relationships it initiates has been recurrently assumed, little empirical evidence is provided in existing scientific literature. Accordingly, a Pearson correlation analysis is performed based on the data presented in the preceding section. The productivity of a complementor is treated as the independent variable, and the number of interfirm relationships it initiated as the dependent variable. As shown in Table 4, there is a significant positive correlation *(0.131)* between productivity and embeddedness, meaning that growth in both variables coincides. Explanations for the mild correlation can be sought in the amount of entrants that already established interfirm relationships in before joining the ecosystem, the relative immaturity of the platform, or the presence of multi-homers in the ecosystem [22].

These findings provide confirmatory evidence for observations from our study of the Google Apps ecosystem [9]. Performing a similar correlation analysis caused us to obtain a significant positive correlation between the productivity of a complementor and its embeddedness. In line with empirical evidence it is

Table 4. Overview of Pearson correlations between number of applications developed and number of interfirm relationships per complementor

Correlations

		# of complements	# of relationships
# of complements	Pearson Correlation	1	.131**
	Sig. (2-tailed)		.002
	N	550	550
# of relationships	Pearson Correlation	.131**	1
	Sig. (2-tailed)	.002	
	N	550	550

**. Correlation is significant at the 0.01 level (2-tailed).

therefore postulated that productivity of a complementor and its embeddedness will be positively related.

Proposition 1: *The number of applications developed by a complementor will be positively related to the number of interfirm relationships it initiates.*

5.2 Partnership Model Participation

In prior work, a platform owner has been argued to be able to actively govern or coordinate its ecosystem. The partnership model has been proposed as a locus of control to enable complementors to more actively participate in the ecosystem; hereby ensuring 'coherent' complementor productivity [11,3,16]. By means of their partnership model, Microsoft can foster lock-in effects [16], platform exclusivity [4] and (quality) control through certification [15]. Ecosystems characterized by fringent governance have been argued to display greater interconnectivity [17,19]. In summary, one could hypothesize that by means of their partnership model, Microsoft can foster the productivity and degree of interaction among ecosystem inhabitants.

To examine whether this argument holds (i.e. Microsoft partners are more actively developing applications and initiating interfirm relationships compared to non-partners), an independent samples T-test is performed. Office365 complementors are distributed in two groups based on their recorded partner status. Partner status is coded as a dummy variable, in which *1* corresponds to a status as Microsoft partner and *0* represents non-partners. In total, *278* complementors are acknowledged as Microsoft partners and *272* are categorized as non-partners. The independent sample T-test was performed at the *95% confidence interval*. Group statistics for both tests are summarized in Table 5.

To explore the influence that the date of entry of complementors may have on our test results, the recorded group means of date of entry for Microsoft partners and non-partners are first compared by means of an independent samples T-test. Microsoft partners (*175 complementors*) on average entered the Office365 ecosystem *two* years and *eight* months ago with a rounded standard deviation of *ten* months, non-partners (*175 complementors*) entered *two* years and *seven*

Table 5. Group statistics for Microsoft partner and non-partner categories

Variable	Partner status	N	Mean	Std. Deviation
# of relationships	Partner	278	1.192	2.925
	Non-partner	272	0.522	2.509
# of complements	Partner	278	2.313	3.351
	Non-partner	272	2.063	3.390

months ago with a rounded standard deviation of *one* year. Based on these group means, there is no significant difference between the date of entry for Microsoft partners and non-partners; $t(350) = 0.593$, $p = 0.728$.

A Microsoft partner in the Office365 ecosystem on average initiated *1.192* interfirm relationships with a standard deviation of *2.925*, a non-partner on average had *0.522* ties with a standard deviation equal to *2.509*. The partner category has a more skewed distribution of interfirm relationships compared to the non-partners category. Based on these figures, there is a significant difference in initiated interfirm relationships between partners and non-partners; $t(538) = 2.895$, $p = 0.004$. Results suggest that Microsoft partners in the Office365 ecosystem have significantly more interfirm relationships compared to non-partners and that lock-ins thus reflect in the network density of the ecosystem.

Proposition 2: *Fostering complementor lock-ins will be positively related to the network density of a proprietary platform ecosystem.*

Another independent samples T-test is performed to investigate the relationship between partnership model participation and productivity. Microsoft partners on average develop *2.313* applications with a standard deviation of *3.351*, and non-partners develop *2.063* applications with standard deviation of *3.390*. Based on these group means, there is no significant difference in development activity between partners and non-partners; $t(548) = 0.871$, $p = 0.384$. Results imply that Microsoft partners are not significantly more committed to development of complementarities compared to non-partners, the small difference in group means may be attributed to random variation or luck.

Proposition 3: *Fostering complementor lock-ins will not influence the productivity of a proprietary platform ecosystem.*

These findings provide preliminary empirical support for the proposed relationship between ecosystem health management, or platform governance, and network density [13,14,11,3,15,16]. The results imply that a platform owner can foster the network density of its ecosystem through its partnership model. Meanwhile, the absence of a relationship between partnership model participation and complementor productivity is surprising and less evident in prior literature. Studies on ecosystem health preservation [13,14] and partnership models [15,16] have suggested that a platform owner can foster the productivity of complemen-

tors. However, empirical evidence provides stronger support for recent findings by Boudreau [12], who based on an empirical study of app stores found that complementors are unlikely to move beyond their development scope, regardless of the strategic incentives offered.

6 Discussion

The results presented in this paper were based on data collected from the Office365 app store, CrunchBase and directly from websites of complementors. Despite all precautions taken, as with any exploratory research this study has a number of limitations. Inherent in the problem domain is the reliance on proprietary sources. A lack of transparency becomes evident when complementors indicate to engage in interfirm relationships, but omit to list their actual partners. Accordingly, the partner activity of such a company is neglected, while in practice it might be embedded in the ecosystem. To limit the influence of this observed threat, interfirm relationships were treated as symmetric ties to increase overall coverage. Furthermore, a preliminary validation of the data collection method in previous research [9] brought it forward as accurate and complete in capturing interfirm relationships in the ecosystem.

Using static analysis for networks that are likely to change over time, poses another limitation for this research. While this threat needs to be acknowledged, previous research [17] showed that the network structure of proprietary ecosystems remained remarkably stable over a 12-year period. Moreover, a remark needs to be placed related to the scope of interfirm relationships, which is likely to span wider than the boundaries of a platform ecosystem. Accordingly, complementor websites rather than alliance databases were used as data sources to include relationships at a finer level of granularity.

Through the presented visualization and analysis of the Office365 ecosystem, this research adds to a limited body of knowledge on the intersection of business or software ecosystems and industry platforms. Being among the first to have presented an ecosystem analysis of a proprietary platform ecosystem, this research introduced a complementary perspective on software platforms and the management thereof. A perspective pertained beneficial to ecosystem managers, and deemed important to the inquiry into platform governance by academia.

7 Conclusion

This paper presented a description and analysis of the Office365 ecosystem. By means of app store data extraction, metadata about complementors was retrieved, after which the interfirm relationships among them were manually identified based on partner mentions on complementor websites and CrunchBase. Subsequent network analysis and bivariate statistical inference oriented itself at investigating the influence of fostering complementor lock-ins through a partnership model on the network structure of a proprietary platform ecosystem and the productivity of its inhabitants.

The Office365 ecosystem was populated by 550 complementors that together developed 1204 applications. Complementors initiated 787 interfirm relationships, which amounted for an average of 1.43 connections per complementor. While 70.18% of complementors was solely connected to Microsoft, the remainder of the ecosystem appeared densely connected. Statistical inference evidenced a positive relationship between the productivity of a complementor and the number of interfirm relationships it initiates. Complementors that participated in the Microsoft Certified Partner Network were found to initiate significantly more interfirm relationships than non-partners. Based on this finding, it was postulated that complementor lock-ins will be positively related to the network density of an ecosystem. Meanwhile, there was no significant difference in productivity between Microsoft partners and non-partners; their productivity remained unaffected and narrow, implying that complementors are unlikely to move beyond their development scope regardless the triggers provided by the platform owner.

This paper provided a step towards better understanding of the effects of enforcing platform governance by examining the influence of partnership model participation on network structure and complementor productivity. Future research should further address the tension between platform governance and the productivity, and embeddedness, of complementors. The results presented in this paper may be extended by means of a more fine-grained quantitative analysis, in which the measurement of platform governance is operationalized in multiple variables to enable multivariate statistical inference. Such an approach would also allow to control for several contextual factors, such as the date of entry into the ecosystem that was mentioned in our study. The method laid out in this paper may also function as a blueprint for longitudinal replication. Another avenue for future research pertains the study of entrance and exit strategies of complementors. Furthermore, this ongoing research project aims to contrast proprietary platform ecosystems to uncover characteristic similarities and differences.

References

1. Gawer, A.: Platform dynamics and strategies: From products to services. In: Gawer, A. (ed.) Platforms, Markets and Innovation, pp. 45–76. Edward Elgar Publishing, Cheltenham (2009)
2. Rochet, J.C., Tirole, J.: Platform Competition in Two-Sided Markets. Journal of the European Economic Assosication 1(4), 990–1029 (2003)
3. Jansen, S., Brinkkemper, S., Finkelstein, A.: Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems. In: Proceedings of the First International Workshop on Software Ecosystems, pp. 34–48 (2009)
4. Boudreau, K.J., Hagiu, A.: Platform rules: Multi-sided platforms as regulators. In: Gawer, A. (ed.) Platforms, Markets and Innovation, pp. 163–191. Edward Elgar Publishing, Cheltenham (2009)
5. Boudreau, K.J.: Open Platform Strategies and Innovation: Granting Access vs. Devolving Control. Management Science 56(10), 1849–1872 (2010)
6. Zhu, F., Iansiti, M.: Entry into platform-based markets. Strategic Management Journal 33(1), 88–106 (2012)

7. Scott, J.: Social Network Analysis: A Handbook, 2nd edn. Sage Publications, Inc., Gateshead (2000)
8. Hanneman, R.A., Riddle, M.: Introduction to social network methods. University of California, Riverside, CA (2005)
9. van Angeren, J., Blijleven, V., Jansen, S., Brinkkemper, S.: Complementor Embeddedness in Platform Ecosystems: The Case of Google Apps. In: Proceedings of the Seventh International Conference on Digital EcoSystems and Technologies, pp. 37–42 (2013)
10. West, J.: How Open is Open Enough? Melding Proprietary and Open Source Platform Strategies. Research Policy 32(7), 1259–1285 (2003)
11. Gawer, A., Cusumano, M.A.: How companies become platform leaders. MIT Sloan Management Review 49(2), 28–35 (2008)
12. Boudreau, K.J.: Let a Thousand Flowers Bloom? An Early Look at Large Numbers of Software App Developers and Patterns of Innovation. Organiation Science 23(5), 1409–1427 (2012)
13. Iansiti, M., Levien, R.: Strategy as Ecology. Harvard Business Review 82(3), 68–78 (2004)
14. 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)
15. van Angeren, J., Kabbedijk, J., Jansen, S., Popp, K.M.: A Survey of Associate Models used within Large Software Ecosystems. In: Proceedings of the Third International Workshop on Software Ecosystems, pp. 27–39 (2011)
16. van Angeren, J., Kabbedijk, J., Popp, K.M., Jansen, S.: Managing software ecosystems through partnering. In: Jansen, S., Brinkkemper, S., Cusumano, M.A. (eds.) Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, pp. 85–102. Edward Elgar Publishing, Cheltenham (2013)
17. Iyer, B., Lee, C.H., Venkatraman, N.: Managing in a Small World Ecosystem: Some Lessons from the Software Sector. California Management Review 48(3), 28–47 (2006)
18. Basole, R.C.: Visualization of interfirm relations in a converging mobile ecosystem. Journal of Information Technology 24, 144–159 (2009)
19. Schilling, M.A., Phelps, C.C.: Interfirm Collaboration in Networks: The Impact of Large-Scale Network Structure on Firm Innovation. Management Science 53(7), 1113–1126 (2007)
20. Freeman, L.C.: Centrality in social networks conceptual clarification. Social Networks 1(3), 215–239 (1979)
21. Watts, D.J., Strogatz, S.H.: Collective dynamics of small world networks. Nature 393, 440–442 (1998)
22. Burkard, C., Widjaja, T., Buxmann, P.: Software ecosystems. Business and Information Systems Engineering 4(1), 41–44 (2012)
23. Gawer, A., Henderson, R.: Platform owner entry and innovation in complementary markets: Evicdence from Intel. Journal of Economics and Management Strategy 16(1), 1–34 (2007)
24. Blondel, V.D., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics 5(10) (2008)

Ecosystem-Driven Software Development: A Case Study on the Emerging Challenges in Inter-organizational R&D

Helena Holmström Olsson[1] and Jan Bosch[2]

[1] Department of Computer Science, Malmö University, Malmö, Sweden
helena.holmstrom.olsson@mah.se
[2] Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden
jan.bosch@chalmers.se

Abstract. Most companies today experience a situation in which they are part of a complex business ecosystem of stakeholders that influence business outcomes. Especially for companies transitioning from selling products to becoming systems, solutions and services providers, this is causing a significant shift in their business strategies and relationships. Instead of focusing on internal processes, companies need to strategically position themselves in a dynamic network of actors to accelerate synergies and value co-creation. However, while this shift in business strategy is inevitable, it is not without challenges. An understanding for how to align internal, as well as external processes is critical, as well as a careful assessment on how to establish strategic partnerships in a dynamic network of interests. Based on on-going research, this paper outlines the emerging challenges that most software development companies face when adopting an ecosystem-driven approach, and the different mitigation strategies to manage these.

Keywords: Company ecosystems, software ecosystems, ecosystem-driven development, challenges.

1 Introduction

The environment in which companies operate is becoming increasingly complex in nature. What we see is a transition from a predominantly intra-organizational perspective to a predominantly inter-organizational perspective where innovation is moving out from the organizational boundaries and where networks of stakeholders co-create value and engage in what has become known as co-opetition [1, 2]. Often co-opetition takes place when companies in the same market domain work together in the exploration and development of new knowledge and products, at the same time as they compete for market-share in the exploitation of the knowledge created. As recognized by Messerschmidtt and Szyperki [3], more and more software companies open up their products for functional extensions by third party developers, allowing for value creation that goes beyond company boundaries [4]. In doing this, these companies need to shift their attention from focusing on internal process efficiency, to

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 16–26, 2014.

how to strategically align with, amplify, and accelerate in synergy with the ecosystem of which it is part. To offer access to external stakeholders rather than being exclusive, to manage the dynamics and dependencies within a large network of stakeholders, and to continuously assess different needs and drivers among stakeholders become critical activities. In the transition towards an ecosystem-driven approach to development, the understanding of how to position oneself in the ecosystem, and the implications this has on business models and strategies becomes increasingly important. Also, there is a need for companies to understand how to align their in-house, and often agile, practices with external stakeholders, such as e.g. suppliers, that typically operate using more traditional ways-of-working.

In this paper, we present findings from an on-going case study in which we work closely with a large company within the embedded systems domain. The company develops products in which hardware is critical, but in which software is becoming increasingly important. Our study focuses on the challenges they face when adopting an ecosystem-driven approach to development, and identifies mitigation strategies to manage these. Our research questions are the following:

- What are the challenges that emerge when adopting an ecosystem-driven approach to software development?
- What mitigation strategies can be applied in order to manage these challenges?

The paper is organized as follows. In section 2 we describe software ecosystems and the reasons for companies to adopt an ecosystem-driven approach to software development. In section 3 we present the research site and the case study method that we use. In section 4, we present the findings concerned with the challenges that emerge when transitioning towards an ecosystem-driven approach to development. Also, we identify mitigation strategies to manage these challenges. In section 5, we discuss our findings and in section 6 we conclude the paper.

2 Software Ecosystems

The emergence of ecosystems is one of the most significant developments for software companies, and it is a topic that recently has attracted significant attention in the software engineering research community [5, 6, 7]. The adoption of a software ecosystem causes companies to open up its successful products and product lines for functional extensions by third party developers [3]. Rather than being exclusive and closing off the product for external developers, organizations are exploring different ways to offer access for external stakeholders without sacrificing important system properties or loosing out on business opportunities or customer relationships. This phenomenon is well elaborated upon in the field of political science and economics, and is described as a situation in which companies interact with partial congruence of interests [8]. As described in this research, companies cooperate with each other to create more value than what they had been able to achieve without the collaboration.

As recognized by Bosch [9], ecosystems take various forms. Typically, a distinction is made between commercial and social ecosystems. In a *commercial ecosystem* the actors are businesses, suppliers and customers, the factors are goods and services and the transactions include financial transactions, but also information and knowledge sharing, inquiries, pre- and post-sales contacts, etc. This is in line with what Moore defined as a business ecosystem that includes suppliers, lead producers, competitors, and other stakeholders [10]. Over time, they coevolve their capabilities and roles, and align themselves with the directions set by one or more key stakeholders. According to Moore [10], the concept of a business ecosystem enables members to move toward shared visions, and to align their investments and find mutually supportive roles.

A software ecosystem is usually defined as a commercial ecosystem consisting of a set of software solutions that enable, support and automate the activities and transactions by the actors [9]. In a software ecosystem there is typically a company providing a software platform and a community of external developers providing functionality that extends the basic platform, and/or users either actively or passively contributing with knowledge, content, goods and services, connections or behavior to the ecosystem. A software ecosystem is often complemented with a *social ecosystem* consisting of users, their social connections and the exchange of information. Here, users either actively or passively contribute knowledge, content, goods and services, connections or behavior to the community and, consequently, to the company providing the ecosystem platform. Successful social ecosystems usually capitalize on the contributions of the users in the to create more value to each individual user as the person provided in the first place. This can be achieved through aggregation and consequent presentation of data uniquely relevant to the user, and through advanced mechanisms that allow for customization.

As can be seen in previous research [9], there are a number of reasons explaining why an ecosystem-driven approach has become so attractive, and why companies adopt this approach to development:

- To increase value of the core offering to existing customers and users.
- To increase attractiveness for new customers and users.
- To decrease costs for commoditizing functionality by sharing maintenance costs with other stakeholders.
- To accelerate innovation through open innovation in the ecosystem.
- To increase collaboration with partners in the ecosystem and share costs of innovation.
- To “platformize” functionality developed by partners in the ecosystem.

As a fairly recent development within the software engineering domain, the concept of software ecosystems reflects a shift in focus from the internals of an organization towards the external environment and the relations and dynamics within this. This introduces a number of opportunities in terms of collaboration, knowledge sharing and value creation, as well as a set of challenges in terms of positioning, ownership and strategic alignment.

In our research, we focus on the challenges that face companies when adopting an ecosystem-driven approach to software development. As recognized by Bosch [11], the company providing the platform needs to go through a significant change in culture in that rather than being the one building products for customers, the role is now to enable external stakeholders to do this. To manage this change without loosing revenue, nor alienating the internal development organization, is a challenging task that requires additional research.

3 Research Site and Method

3.1 Research Site

This paper presents on-going research based on a case study conducted at a company within the embedded systems domain. Our case company is world leading in network video and offers products such as network cameras, video encoders, video management software and camera applications for professional IP video surveillance. At the moment, the company is experiencing a shift in focus, i.e. from being predominantly a product manufacturer the company is adding a focus on systems, solutions and services. This shift is causing the company to carefully consider the ecosystem of which it is part, and define alternative business strategies involving a broad spectrum of stakeholders in this network. The strategies focus on how to forward integrate in the value chain, how to complement the existing product portfolio with service offerings and, finally, how to adopt an ecosystem-driven approach to software development in which customers, suppliers, competitors and end-customers engage in joint innovation efforts and value co-creation.

The main motivation for the company to engage in this research is the increasing need to understand the ecosystem stakeholders, their drivers, their needs and their challenges in order to strategically align, amplify and accelerate in synergy with these.

3.2 Research Method

Our paper reports on an on-going single case study [12, 13] involving a large company in the embedded systems domain. The project was initiated in October 2013 with the first data collection activities starting in November 2013. The main data collection method used is semi-structured group interviews with open-ended questions where the researchers meet with groups of four to seven people for a two to three hour session. Also, the two researchers continuously meet with key stakeholders to discuss project activities, project findings, and how to best accelerate the adoption of an ecosystem-driven approach within the company. These meetings are conducted in a workshop style with a mix of presentation and discussion from researchers and practitioners.

So far, one group interview and three workshop meetings have been held at the company. For the group interview, one of the researchers met with a group of seven

people representing global partners and business development, product and segment marketing, product maintenance, global sales, product management, and core technologies. In addition, two company representatives that work as contact persons towards the research project, and with roles such as project and product managers, were present to listen in and to be part of the introduction in which the research focus and topic was explained to the group. The group interview lasted for three hours. Notes were taken during the interview and these were shared between the two researchers to allow for a discussion regarding the interview session, the answers that had been given and the overall impression of the discussion. Also, the participants were asked to provide their input on a number of topics that were listed and handed out them during the workshop. This allowed the researcher to have documentation consisting of both personal notes, but also notes taken by the participants themselves. Finally, the group interview was recorded. The workshop meetings involved key stakeholders in the company with an interest in the topic and with the intention to have people from their unit participate in up-coming research activities. The workshop meetings were one to two hour sessions in which the two researchers and the company representatives all contributed with content and items for discussion. In total, three workshop sessions have been conducted, involving six people with roles such as project and product managers, and identified as key stakeholders for this project.

In terms of data analysis, an interpretive approach was adopted as described by Walsham [14]. While this approach has similarities with the qualitative grounded theory approach [15], it is not as strict in its coding process. Rather, the researcher documents his or her impressions during the research, for example after each interview, in order to generate more organized sets of themes after a group of interviews or a major field visit. When having this organized set of themes, the researcher then carefully reflects on what can be learnt, and what implications can be drawn, from the field data [14].

A problem that has been identified in relation to qualitative research is that different individuals may interpret the same data in different ways [16]. This problem is addressed in two ways. First, the grounded theory method prescribes coding processes that provide a traceable, documented justification of the process by which conclusions are reached. Second, we use a 'venting' method, i.e. a process whereby interpretations are continuously discussed with professional colleagues [17]. By sharing notes, and by discussing the results of group interviews and workshops, we develop an accurate understanding of the company and the challenges it faces.

4 Findings

In this section, we present findings from our on-going research. We identify the emerging challenges that face companies when adopting an ecosystem-driven approach to development. Also, we present mitigation strategies for how to manage these challenges. The challenges and the mitigation strategies were all identified

during a group interview and workshop meetings at the case company, and they reflect concerns critical for successful R&D management in an increasingly complex business environment.

4.1 Challenges

Our group interview and the workshops at the company reveal a number of challenges that emerge when adopting an ecosystem-driven approach to development.

A recurrent theme, and the major challenge, is the difficulty to identify rewarding business models that add revenue, at the same time as it allows for external stakeholders to be value co-creators. The concern that our interviewees express is how to minimize opportunities for their competitors, while at the same time maximize the capacity of the ecosystem. One risk that was mentioned during the group interview was a scenario where competitors could potentially provide the same value for free, and monetize somewhere else. In this discussion, the interviewees all agreed that one critical success factor is how to support collaboration in the ecosystem without becoming the exchangeable component, and to find ways to share core competencies without loosing business opportunities. Also, the interviewees identified the need to understand what development should be made in-house and what development should be outsourced to external stakeholders in the ecosystem. Both these concerns relate to the challenge in finding an appropriate business model that drives revenue at the same time as it fosters ecosystem collaboration.

A second challenge is the intention in the company to get closer to its end-customers. Today, there are a number of intermediaries such as distributors and installers, and the interviewees experience a situation in which important feedback from customers is lost in the complex hierarchy of stakeholders. However, moving closer to customers, i.e. forward integrate in the value chain, might upset other stakeholders who already have a close relationship to end-customers, and do not want to share, or even loose this. To manage this situation, the interviewees expressed a need to understand more about the drivers for different stakeholders, and what motivates them to be part of the ecosystem.

Third, the company is moving from a transactional business model in which box products are the main focus of attention, towards a relationship-based business model in which the company provides customers with entire systems, solutions and services. This is a major transformation for the company, and it implies new relationships and dependencies to stakeholders in the ecosystem. The interviewees express concerns for not having all the necessary skills for achieving this, and they see challenges in monitoring end-to-end service delivery to new stakeholders in the ecosystem.

In table 1 below, we summarize the challenges that were identified by our interviewees as emergent when adopting an ecosystem-driven approach to software development:

Table 1. The emerging challenges that companies face when adopting an ecosystem-driven approach to software development

Challenges:	**Concerns:**
Identify a rewarding business model	• How to minimize opportunities for our competitors while at the same time maximize the capacity of the ecosystem? • How to support collaboration without becoming the exchangeable component? • How to share core competence without loosing business opportunities? • What development should be made in-house and should be outsourced to other stakeholders in the ecosystem?
Forward integration in the value chain	• How to move closer to end-customers without upsetting other stakeholders in the ecosystem? • What are the drivers of the different stakeholders that constitute the ecosystem?
Transition towards systems, solutions and services	• What new relationships and dependencies emerge to other stakeholders in the ecosystem? • How to monitor end-to-end service delivery to new stakeholders in the ecosystem?

4.2 Mitigation Strategies

In order to manage the challenges as identified above, our study reveals a number of possible mitigation strategies.

As the most rewarding, and long-term strategy, our interviewees identify the need to form strategic partnerships with stakeholders in the ecosystem. This is considered critical in order to mitigate competition and instead foster collaboration among key stakeholders with potentially competing interests. In particular, this strategy is critical to mitigate the challenge in identifying a rewarding business model. As opposite strategies to this, our interviewees discuss the possibility to build more proprietary solutions that make their products less open and less exposed to competition. However, while this strategy would allow the company to charge a license fee from any other stakeholder who wants to use the proprietary product platform, it would be contrary to the ecosystem-driven approach and the use of open and standardized

solutions. As a mitigation strategy, this discussion reflects a short-term solution rather than a long-term solution, with the risk to create more problems than it solves.

A second mitigation strategy is to become the preferred component supplier and use offered solutions and services to drive revenue. This strategy is recognized as one way to address the challenge associated with forward integration since it would allow the company to establish closer relationships with stakeholders with whom they have previously not interacted. In adopting a solution and services provider role, this strategy would enable closer contact with end-customers.

Finally, our interviewees emphasize the need for qualified training for existing employees as a strategy to mitigate the challenge with transitioning towards systems, solutions and services. In order for the company to be successful in this transition, functions besides the development organization need to be available and supportive of the new business opportunities offered within the ecosystem. As one way to realize this, our interviewees suggest rotation within the company to broaden the skillset among existing employees.

In table 2 below, we summarize the mitigation strategies that were identified to manage the challenges that emerge when adopting an ecosystem-driven approach to software development:

Table 2. Mitigation strategies to manage challenges that emerge when adopting an ecosystem-driven approach to software development

Challenge:	**Mitigation strategy:**
Identifying a rewarding business model	• Form strategic partnerships with key stakeholders in the ecosystem
Forward integration in the value chain	• Become the preferred component supplier • Adopt a solution and services provider role
Transition towards systems, solutions and services	• Qualified training for existing employees • Rotation within the company for existing employees

5 Discussion

Adopting an ecosystems approach by shifting focus from the internals of the organization towards its external environment increases the value of core offerings to existing customers, increases attractiveness for new customers, decreases costs by sharing maintenance costs, and accelerates innovation through co-creation of value [3, 6, 9]. While this is a general trend in the software industry [18], and associated with a number of opportunities [6], it comes with a set of challenges that need to be mitigated in order to fully capitalize on the potential inherent in the ecosystem.

Our case company experiences a complicated business ecosystem of stakeholders that influence their business outcomes. As recognized in previous research [3, 4], the

business ecosystem is becoming increasingly important to companies due to a variety of factors including the outsourcing of parts of the development, the transition from products to systems, solutions and services, and the increasingly open nature of innovation where networks of companies and customers co-create value. Especially, the transition towards systems, solutions and services causes a significant shift in how companies interact with other stakeholders in the ecosystem. An increasing emphasis is put on strategically align with, amplify, and accelerate in synergy with other stakeholders in the ecosystem.

In our on-going study, we work closely with a company within the embedded systems domain that is adopting an ecosystem-driven approach to software development. In doing this, the company is shifting attention from an intra-organizational focus to an inter-organizational perspective where value creation is moving out from the organizational boundaries to include external stakeholders and customers [6].

Our study identifies a number of challenges that the company faces when adopting an ecosystem-driven approach to software development. As the major challenge, our interviewees identifies the difficulty to identify rewarding business models that add revenue to the company, at the same time as it allows for external stakeholders to be value co-creators. Although previous research on software ecosystems has primarily reported on new business opportunities [6], our study reveals a number of concerns associated with monetization on business offerings. One of the concerns that our interviewees express is how to minimize opportunities for their competitors, while at the same time maximize the capacity of the ecosystem. To do this, mitigation strategies that drive revenue at the same time as they allow ecosystem collaboration need to be implemented. As the most rewarding and viable strategy, our interviewees identify the need to form strategic partnerships with key stakeholders in the ecosystem.

A second challenge experienced in our case company is the intention to forward integrate in the value chain and move closer to the end-customers. While this opens up for shorter feedback loops and an increased understanding for customer needs [19, 20], it might upset other stakeholders who already have that close relationship to end-customers. As highlighted in our study, becoming the preferred component supplier and use offered solutions and services to drive revenue can help mitigate this challenge. While challenging to implement, this strategy allows companies to establish relationships with stakeholders with whom they have previously not interacted. To do this, an understanding for different stakeholders' drivers and motivators is critical to not upset existing relationships.

Finally, our study identifies the challenge with transitioning from a transactional business model towards a relationship-based business model where companies provide customers with systems, solutions and services instead of box products. This is a major transformation, and it implies new relationships and dependencies to other stakeholders in the ecosystem. The interviewees express concerns for not having all the necessary skills for achieving this. To mitigate this challenge there is the need to instill the skills and attitudes needed for creative enterprise and foster open innovation characterized by a culture of healthy risk-taking and collaborative activity. To

mitigate this challenge, qualified training for existing employees is critical and rotation of employees can be one way to realize this.

While our study reflects the experiences from a single company, we believe that the challenges and the mitigation strategies we identify are relevant to other companies in the software development domain when adopting an ecosystem-driven approach to development. Based on our case study findings, we see that if successfully engaging the ecosystem stakeholders in the product development process, companies can empower innovation, foster creative enterprises, unleash open markets characterized by healthy risk-taking and finally, facilitate efficient creation, circulation and diffusion of knowledge.

6 Conclusions

In this paper, we present findings from an on-going case study conducted in a company within the embedded systems domain. Based on a group interview and workshop meetings with key stakeholders, we: (1) identify emerging challenges that companies face when adopting an ecosystem-driven approach to development, and (2) identify mitigation strategies to manage these challenges. The challenges and the mitigation strategies reflect concerns critical for successful R&D management when adopting an ecosystem-driven approach to software development.

Concluding, in being a fairly recent development within the software engineering domain, the concept of software ecosystems reflects a shift in focus from the internals of an organization towards its external environment. So far, this shift has received limited attention even though its adoption is now accelerating by increasingly many companies. With this research, we identify challenges and mitigation strategies that we believe will be of relevance for a wide range of companies adopting an ecosystem-driven approach to software development.

Acknowledgements. This study was funded by Malmö University as part of a research collaboration between Malmö University and the Software Center at Chalmers University of Technology and University of Gothenburg, Sweden. We would like to thank the company involved in the study for the time and engagement allocated by all interviewees.

References

1. Bengtsson, M., Kock, S.: Coopetition in Business Networks – to Cooperate and Compete Simultaneously. Industrial Marketing Management 29, 411–426 (2000)
2. Dagnino, G.B., Padula, G.: Coopetition strategy: towards a new kind of interfirm dynamics for value creation. In: EURAM 2nd Annual Conference, Stockholm School of Entrepreneurship, Sweden, May 8-10 (2002)
3. Messerschmitt, D.G., Szyperski, C.: Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press, Cambridge (2003)

4. 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), 67–76 (2010)
5. Hanssen, G.K.: A Longitudinal Case Study of an Emerging Software Ecosystem: Implications for practice and theory. Journal of Systems and Software 85(7), 1455–1466 (2012)
6. Jansen, S., Finkelstein, A., Brinkkemper, S.: Sense of Community: A research agenda for software ecosystems. In: Proceedings of the 31st International Conference on Software Engineering, Companion Volume, pp. 187–190. IEEE (2009)
7. 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), 1467–1478 (2012)
8. Asaro, V.F.: Universal Co-opetition: Nature's Fusion of Cooperation and Competition (2011) ISBN 978-1-936332-08-3
9. Bosch, J.: From Software Product Lines to Software Ecosystems. In: Proceedings of the 13th International Software Product Line Conference (SPLC), San Fransisco, CA, USA, August 24-28 (2009)
10. Moore, J.F.: The Death of Competition: Leadership & Strategy in the Age of Business Ecosystems. Harper Business, New York (1996) ISBN 0-88730-850-3
11. Bosch, J.: Software Ecosystems: Taking software development beyond the boundaries of the organization. Journal of Systems and Software 85(7), 1453–1454 (2012)
12. Walsham, G.: Interpretive case studies in IS research: Nature and method. European Journal of Information Systems 4, 74–81 (1995)
13. Runesson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2009)
14. Walsham, G.: Doing Interpretive Research. European Journal of Information Systems (15), 320–330 (2006)
15. Corbin, J., Strauss, A.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Sage, California (1990)
16. Kaplan, B., Duchon, D.: Combining qualitative and quantitative methods in IS research: A case study. MIS Quarterly 12(4), 571–587 (1988)
17. Goetz, J., LeCompte, D.: Ethnography and Qualitative Design in Educational Research. Academic Press, Orlando (1984)
18. Qualman, E.: Socialnomics: How Social Media Transforms the Way We Live and Do Business. John Wiley & Sons (2009)
19. Holmström Olsson, H., Bosch, J.: Post-Deployment Data Collection in Software-Intensive Embedded Products. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 79–89. Springer, Heidelberg (2013)
20. Holmström Olsson, H., Bosch, J.: Towards Data-Driven Product Development: A Multiple Case Study on Post-Deployment Data Usage in Software-Intensive Embedded Systems. In: Fitzgerald, B., Conboy, K., Power, K., Valerdi, R., Morgan, L., Stol, K.-J. (eds.) LESS 2013. LNBIP, vol. 167, pp. 152–164. Springer, Heidelberg (2013)

Why Early-Stage Software Startups Fail: A Behavioral Framework

Carmine Giardino, Xiaofeng Wang, and Pekka Abrahamsson

Free University of Bolzano, piazza Domenicani 3 39100 Bolzano, Italia
computer.science@unibz.it
http://unibz.it

Abstract. Software startups are newly created companies with little operating history and oriented towards producing cutting-edge products. As their time and resources are extremely scarce, and one failed project can put them out of business, startups need effective practices to face with those unique challenges. However, only few scientific studies attempt to address characteristics of failure, especially during the early-stage. With this study we aim to raise our understanding of the failure of early-stage software startup companies. This state-of-practice investigation was performed using a literature review followed by a multiple-case study approach. The results present how inconsistency between managerial strategies and execution can lead to failure by means of a behavioral framework. Despite strategies reveal the first need to understand the problem/solution fit, actual executions prioritize the development of the product to launch on the market as quickly as possible to verify product/market fit, neglecting the necessary learning process.

Keywords: Software startups, customer development, lean startup.

1 Introduction

Software startups launch worldwide every day as a result of an increase of new markets, accessible technologies, and venture capital [1]. With the term *software startups* we refer to those temporary organizations focused on the creation of high-tech and innovative products[1], with little or no operating history, aiming to grow by aggressively scaling their business in highly scalable markets [2].

New ventures such as *Facebook, Linkedin, Spotify, Pinterest, Instagram, Groupon* and *Dropbox*, to name a few, are examples of startups that evolved into successful businesses. Despite many success stories, many software startups fail before they have fulfilled their commercial potential [3].

Even though startups share some characteristics with similar contexts (e.g. small and web companies), the combination of different factors makes the specific development context quite singular [2,4].

However, failures of startups received little attention [5]. Despite the quick proliferation of startups' communities, they have been able to absorb little more

[1] With the term "product", we refer to both software products and software services.

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 27–41, 2014.

than the basic patterns of how to build a startup [4]. Moreover, more than 90% of startups fail, due primarily to self-destruction rather than competition [6].

This study aims to understand the software startups' dimensions, which impact failed software startups. The failure has been determined from the point of view of their chief executive officers (CEOs), who have broad perspectives on their startup organization [7].

This study was elaborated based on a multiple-case design, implemented with in-depth narratives of the two project cases, covering a wide spectrum of themes and iteratively adjusting the direction of the research according to the emerging evidence.

The results show that the two startups didn't follow consistent strategies to understand the problem they were trying to solve, consequently diluting their focus on running in the wrong direction. Despite conventional dimensions emerged to be important to improve, such as *Market, Team, Product, Business*, strategies and executions of their development were not consistent to the state of their problem/solution fit, presented by means of a behavioral framework. The behavioral framework provides a potential reason for the failure of software startups as the result of the analysis of the extrapolated data.

The rest of this paper is composed as follows: in section 2, background and the related work are presented according to the relevant disciplines. Section 3 presents the empirical research design which is followed by the presentation of the case studies results in section 4. The conclusions are identified and discussed in section 5. The paper is summarized with section 6 addressing the limitations and identifying future research needs.

2 Background and Related Work

In this section knowledge on the definition of software startups and how they differ from established companies is presented according to existing literature. Subsection 2.1 examines characteristics of startup companies. Subsection 2.2 describes how the state-of-the-art presents constraints, which impact the company's survival. Subsection 2.3 presents dimensions to consider when building a startup company.

2.1 Singularity of Software Startups

Startup companies exhibit many characteristics which reflect both engineering and business concerns. Constraints of those characteristics differ from those of established companies [5].

Established companies present advantages with respect to starutps, such as fewer internal communication and coordination problems, a foundation of established products, partners, and customers with a greater shared history and vision [4]. Sutton [4] provides a characterization of software startups, defined by the challenges they face with:

- little or no operating history: startups have little accumulated experience in development processes and organization management.
- limited resources: startups typically focus on getting the product out, promoting the product and building up strategic alliances.
- multiple influences: pressure from investors, customers, partners and competitors impact the decision-making in a company. Although individually important, overall they might not converge to support a clear decision-making.
- dynamic technologies and markets: the newness of software companies often requires them to develop or operate with disruptive technologies[2] to enter into a high-potential target market.

2.2 Failure Assessment

Modern entrepreneurship, born more than thirty years ago [9], has been boosted by the advent of consumer Internet markets in the middle of the nineties and culminated with the notorious dot-com bubble burst of 2000 [10]. Today, with the omnipresence of the Internet and mobile devices, we are witnessing to an impressive proliferation of software ventures- metaphorically referred to as the startup bubble. Easy access to potential markets and low cost of service distribution are appealing conditions for modern entrepreneurs [11]. Inspired by success stories, a large number of software businesses are created everyday. However, the great majority of these companies fail within two years from their creation [3].

Despite the believe that the success rate of startups has the potential to dramatically increase economic growth on global scale [6], very few and controversial findings about their failures have been found by the researchers in the last years [5].

A prominent contributor as researcher and practitioner of the startup community is Steve Blank. In his research [2] he describes how very few startups fail for lack of technology, rather they almost always fail for lack of customers. For a company trying to enter a very innovative market without proof of functionality in the real world there are more chances of failure. Customer feedback is assumed to reduce the perceived risk in a software startup. Especially in the marketing of complex and software-intensive products, experimental knowledge is crucial. However the use of customer feedback by software startup companies is distinctly under-researched [11]. Startup companies have reported that they use the first customer feedback to develop the product further, find arguments for sales and marketing purposes, learn project skills, and study the business logic in their industry [12,13]. The focus of success moves to the abilities of finding the first customers and expanding the business abroad, after saturating the limited size of local markets.

In the course of attracting and keeping customers, Blank suggests a process to place aside to product development, which aims to discover and validate

[2] A new technology that unexpectedly displaces an established technology. It does not rely on incremental improvements to an already established technology, but rather tackles radical technical change and innovation [8].

the right market for an idea. The first part of the discovery consists of finding the right **problem/solution fit**. The aim is to test the riskiest hypotheses of the problem taken in consideration by implementing a first solution. The second step is to build the right product features that solve real customers' needs, also known as the **product/market fit**. If the product/market fit is not achieved, then a problem/solution fit must be reiterated, operation known as *pivoting*. Ultimately, in order to grow, companies have to test their business models, investing capital and executing tactics for acquiring and converting more customers of their potential market.

2.3 Dimensions for the Evaluation of Software Startups

Subsection 2.2 examined the customer development methodology to describe the objectives to scale a business concept. When resources are scarce, survival and success depend most heavily on the executives and managers, who are responsible for shaping, directing, and implementing company strategies. The importance of people in these roles derives from the need to keep the company focused and moving ahead [4]. However the customer development methodology doesn't define the focus in which "ahead" lies. Despite the direction might shift continually due to the dynamic and unpredictable context in which startups operate [5], some dimensions at all stages of the life cycle remain crucial.

Draw upon the study of MacMillan et al. [14], applied in startup contexts, four holistic dimensions are taken in consideration. The **team** as the core element. Software project managers have long recognized the importance of good people for successful development [5]. New software companies are often established to develop a technologically innovative **product** [4]. Yet, the **business** and the way it evolves can set the company growth and its place in the market [15]. The uncertainty of markets inevitably brings financial risks, which on the other hand are fundamental to set-up any company. Ultimately knowing the **market** is essential to evaluate the needs of the final customers [2].

3 Research Approach

The goal of this study aims to understand the software startups' dimensions which affect failure in a company. It is achieved through investigating how project goals are defined and decision-making is executed in the period of time that goes from idea conception to the validation of a product on the market. The following section presents the research design and rationale for methodology selection. Subsection 3.1 presents the context of the studied software startups and the short description of their business ideas.

Following a systematic mapping study (SMS) [16], we conducted an exploratory multiple-case study [17]. We executed two semi-structured interviews (with the CEOs of two failed startup companies) integrated with narrative descriptions of their failures. Moreover data were triangulated with external documentation, collected and elaborated by the founding teams. From all the data, we extracted and analyzed a model explaining the failure phenomenon of these two startups.

A first preliminary SMS in the software engineering (SE) databases revealed a quite wide gap in studies addressing failure in software startups, but at the same time it showed us how broad the domain is. A definition which confirms the suitability of this methodology is provided by Kitchenham et al. in [18]: SMSs ''are designed to provide a wide overview of a research area, to establish if research evidence exists on a topic and provide an indication of the quantity of the evidence [...]".

Next a multiple-case study approach was chosen in view of its ability to be flexible in deepening the understanding of a specific problem, covering a wide spectrum of topics and analyzing recurring patterns among them. Going from the beginning extensively into a specific problem in software startups context would not be worth trying because we didn't find a sufficient number of studies that constituted a solid body of knowledge, confirmed by [5]. On the other hand, trying to select a broad research topic would carry the risk of spending a lot of time in achieving a better understating of the problem without being able to provide any significant results. A multiple-case design allows behavior analysis over control analysis, with the benefit of understanding the phenomena in an unmodified setting [17]. It also allows to analyze patterns across many cases and to understand how they are affected by local conditions to develop more sophisticated descriptions and powerful explanations [19].

Moreover, as we wanted the ability to transfer our ideas to startup practitioners, we applied a comparative analysis of the state-of-the-art and the case study findings following evidence-based software engineering principles [20,21] and providing empirical evidences supporting startups' decisions. The sampling of the two startups has been selected according to the features identified in subsection 2.1. Yet they are two failed evidences, as required by our research goal, as perceived from the two CEOs from their idea conception to the first open beta release.

We captured the most relevant aspects of the companies activities, letting emerge patterns from the data and adjusting the framework as we proceeded. The findings are derived primarily from the observations and perceptions of the two CEOs. The reason for selecting CEOs as respondents is that they have experienced all the decision-making process of the companies. Furthermore, they participated in the creation of the companies since their inception, having strong insights into the working activities they conducted and financed. However, the findings are finally compared to the existing literature on software startups' failure to improve generalizability.

In this paper we address the achievements of four dimensions defined by MacMillan et al. [14] (i.e. product, team, business, market).

The whole procedure was executed by the first author. Subsequently reviewed by the second and third authors. When necessary all the authors performed an in-depth review of the design of the study and discussed the findings to improve their validity.

3.1 Background to the Cases

The first startup (called Milkplease[3]) aims to deliver grocery shoppings from local supermarkets to the door-step of final consumers through the collaboration of neighbors. It provides a web application allowing people to know that a neighbor needed to do his shopping.

Milkplease was founded by an engineer and a computer scientist by using their private savings. Running the business for one year, they increased the team size in the last months with three more people, specialized in marketing and service design. The business model expected profits by applying a service cost, in the form of percentage over the total shopping amount for each accomplished delivery. Initially the startup targeted full-time working employees for making orders, and students for making deliveries in a small district of an European city.

The second startup (called Picteye[4]) aims to create an on-line service to sell pictures of public and private events. They provided a web market-place where people could sell and buy pictures. They started with providing pictures of one big event in Italy. Then, they moved to use the application for other big events around Europe.

The overall team grew from 3 software engineers (founders) to 6, covering business and marketing positions. Picteye has been running for two years, trying to launch the product several times without any positive response from the market. The business model expected profits by selling pictures (digitally delivered to private individuals). The startup targeted photographers for uploading pictures, and people participating to events for making orders.

4 Results

By analyzing the interview transcripts and narrative descriptions with the multiple-case study process, we extracted the information to describe the thematic areas of the startups and construct the final framework, presented in subsection 4.1. The thematic areas are divided according to MacMillan et al. dimensions.

This section contains quotes from the CEOs, referred with the capital "C". The trailing number indicates the software startup (e.g. "C1").

The **product** dimension characterizes the software development strategies in the two case studies. Since the two startups wanted to launch the product on the market as quickly as possible, they built an initial prototype and iteratively refined it over time, similarly to the concept of "evolutionary prototyping" [22]. However no initial hypotheses were defined from the business perspectives, and consequently no minimal viable product (MVP), as meant by Ries [23], was achieved.

"We were so excited about the technology that we put it upfront. We wanted the product to be ready for being used by many customers. The day of the launch arrived, but no one made a transaction. We still don't know why". [C1]

[3] Milkplease website is available at `www.milkplease.it`

[4] Picteye website is available at `www.picteye.com`

"I don't know how and why people buy photographs on-line. My startup has not answered to this question either. We were in doing mode. Ultimately 1 photo has been sold, and this was after presenting the platform in a conference, when someone in the audience saw himself in the picture gallery and decided to give the system a try". [C2]

Building a product without pursuing the problem/solution fit hasn't provided any learning process during the first period of the startup creation. In this regards, the learning based approach has been neglected by the two CEOs to benefit their attitude to acquire more and more customers. Yet, testing the initial hypotheses means have better understanding of the product risk, building the right solution for a problem worth solving.

The **team** dimension represents the characteristics of the people involved in building the company. The founding teams of the two startups were initially formed by full-stack engineers, able to tackle different problems at different levels of the technology stack, and covering multi-roles since they all participated in the business and development processes. However entrepreneurial characteristics were missing, such as motivation and ability to evaluate and react to risks.

"We started the project with a clear vision of what we wanted to achieve. However, we soon shifted the vision towards money concerns, refining our business model without having familiarity with the market". [C1] "During the first event week, we were presenting the project for the first time and part of the founding team bailed out. The motivation was missing". [C2]

Motivation has been felt as crucial to pursue the success of the first product release. Indeed, everyone should have been involved to understand customers' expectations and allow the entrepreneurs to evaluate and react to risks perceived during this period according to the collected feedback.

The **market** dimension represents the strategies for customers' acquisition. The customers of the two startups were initially acquired to pursue the product/market fit by using extensive press coverage.

"We participated to different startups' contests, where we got quickly hype. In the beginning we were sponsored by press, indeed many potential customers started visiting our web-page. We wanted all of them starting using the product from day one". [C1] "I organized an event, sending an email to all the people I could. I said few motivational words about the product. However, only few people went to the event, and shadows started growing around the project". [C2]

Customer acquisition started to be the first focus, even though the problem/solution fit was not discovered yet. With the benefit of hindsights, the CEOs argued that having small amount of participants in the beginning of the process would have allowed the startup to build an effective path to customers who care about the promoted solution, and ultimately found a market that would have supported a viable business.

The **business** dimension focuses on strategies to make profits out of the product. From the business perspective the two startups started to focus on maximizing the profit aspect too early, over-planning the model and prematurely adapting the market according to the business, and not vice-versa.

"Money became a first concern. As soon as we realized the product, we wanted to cover all the expenses, and maximize profits". [C1] "After the event, we started waiting the money flow in the form of completed order forms. We had everything ready for the great success". [C2]

The two cases reveal how the several dimensions were executed without regular customers' feedback loop, without adapting the business and the product to a changing market. To understand how the CEOs did not execute the strategies to capture the problem/solution fit, we make use of a behavioral framework (see figure 1).

4.1 The Behavioral Framework

Through analyzing the results, the two startups moved through similar thresholds and milestones of development, which we segmented into stages. From the narratives we wanted to preserve the temporal precision of the events moving along the prominent objectives. We took advantage of hindsights to see what the strategies would have been during the first period of time and what was actually executed.

The final framework (see figure 1) contains the evolution of four dimensions to build the startup, according to their exploration and validation stages. Exploration stage describes the activities focused on solving a meaningful problem, finding a viable solution. The validation stage allows to understand if the product is of interest to the potential market and if customers are willing to buy it.

The actual stage describes what they should have focused on, according to their initial strategies. The behavioral stage describes the execution of different dimensions, inconsistent to what they had initially planned. The time-line suggests how startups would have worked through the different dimensions (further analyzed in section 5).

From the product perspective a first solution should contain the minimal functional features to address the riskiest hypotheses of the business idea. Within an evolutionary approach, user experience should be improved to let customers smoothly use the product. Eventually new functionalities can be added to scale in the major market.

"We wanted to test the product on the market as soon as possible. However we had no clear hypotheses to test. We wanted to be ready for the public launch, even though we didn't evaluate known risks." [C1] "We thought and hoped that the market launch date was coming shortly to finally see the results. We knew all our thoughts were just speculations. However, I had not been thinking about lean startup method, we were entusiastic about the technology". [C2]

From the team perspective, up to when the problem and solution don't fit, the team should have entrepreneurial characteristics to be able to share the vision and evaluate the risks of the market. Only then team should start covering missing roles, such as marketing and business specialized positions.

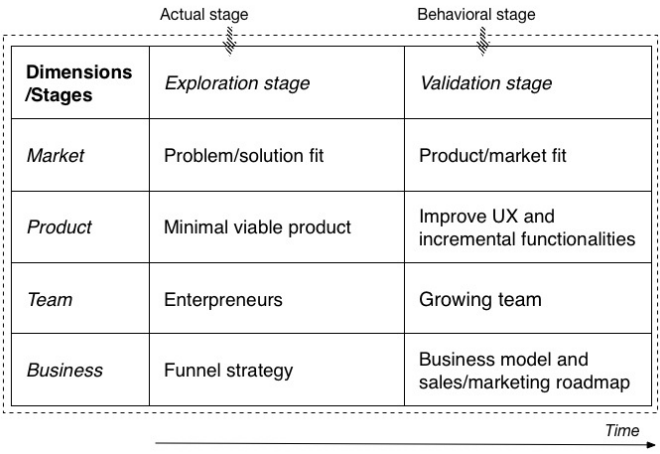

Fig. 1. Behavioral framework

"The vision was not clear as soon as we were ready to test the product. Seemingly the time spent on the product changed our expectations. The team size grew with marketing specialists and service designers". [C1] "No one was fully committed to the project. The team's motivation went down soon. However we started to grow with more specialists". [C2]

From the business perspective at the beginning the two startups should have planned how to converge traffic to satisfy and retain a small group of users. However they focused on getting the business model refined.

"The business model has been polished to maximize profits even before the product was ready. We invested our own money, and wanted them back as soon as possible. When we realized there was no market for what we realized, it was surprisingly too late to change the product again" [C1] "The company was running and it was time to get money. We asked for 20k euro to further develop the idea. We lost every month 100 euro, affordable for the first months. Soon this was not true anymore" [C2]

At the beginning the two startups tried to be profitable, rather than moving along the problem/solution fit and starting covering the need of a funnel strategy[5]. The first objective of getting profits has never been reached. Ultimately no more resources were available for possible pivots. Polishing the product and business model has consumed more resources than moving through a build-measure-learn cycle [23].

The difficulty of covering the negative cash flow, before the product or service brings revenue from real customers, has become critical over time. Up to the first

[5] A strategy that illustrates the customer journey towards the purchase of a product or service, improving the awareness of the existence of a product, activating interest and desire to choose the product [2].

launch no feedback has been given by potential customers, without effectively capturing their needs.

"We ran the system within a group of potential users to get opinions. They helped us only with the UX". [C1] "An amateur photographer would have given us excellent feedback. He did a brief experiment, and sent me an email where he killed the system. I thought he was not the type of customer using the system. I continued waiting for the big launch." [C2]

5 Discussion

This section compares the overall findings of this study to existing literature, in order to emphasize crucial factors of failure in early-stage software startups. Subsection 5.1 provides insights on the lack of problem/solution fit, whilst subsection 5.2 explores the revealed weak learning process. Ultimately subsection 5.3 gives an understanding of how the framework can be adopted by practitioners and researchers.

The two described startups didn't consistently follow the defined stages. Their behavioral stage differed from their actual stage working on some operations prematurely.

As shown in the results instead of targeting the problem/solution fit, they were pursuing the product/market fit, from the *product, team, market and business* perspectives. Aiming at the product/market fit, startups focus on validating a product instead of discovering and testing a problem space. Moreover, where startups should be testing demand for a functional product, they focused on streamlining the product and making their customer acquisition process more and more efficient.

5.1 Lack of Problem/Solution Fit

Trying to validate a product for a problem you haven't encountered yet is a waste. Instead, being able to demonstrate problem/solution fit through discovering what are the needs of your first customers is much more viable than an untested story [24].

The two startups were improving the products' user experience without having tested the MVP first. Since 1994, Carmel [25] reports the need of flexible and rapid development solutions to shorten time-to-market to test assumptions. Eric Ries [23] shows an evolutionary approach to focus on implementing a limited number of suitable functionalities (MVP). The MVP does not require to comply with heavy quality constraints, enabling the team to quickly implement suitable functionalities to test business assumptions, ready to be validated by final users. The evolutionary approach enhances the effectiveness of the product, and enables the company's capabilities to adjust the trajectory of product development [24].

The two case companies showed initial evidences of a lack of systematic feedback from customers to improve their market understanding. However, startups

face uncertain conditions, which should lead to a fast learning from trial and error, with a strong customer relationship. Avoiding wasting time in building unneeded functionalities helps also in preventing exhaustion of resources [26,27,4].

As team dimension is concerned, in order to have the focus and the ability to evaluate risks, Carmel suggests that entrepreneurs need to look for a well formed, skilled core development team, rather than just a set of product ideas and features. Moreover the team must be empowered with full-stack and self-organization settings, paired with the characteristics of real entrepreneurs. Studies suggest that entrepreneurs possess 'special' personality characteristics [28,9]. One of the key determinants of success in startup companies is the passionate behavior of the founders. People who lack passion often use the first barrier they encounter as an excuse for failure. People who have high passion will do whatever it takes to overcome those barriers. "What we can achieve in life depends on a number of things: how hard we work, how smart we work, how much leverage we have on the work we do, and how much courage we have in pursuing our goals. How hard we work is tied to how passionate we are" [29].

In addition to the missing personal motivation, the two startups were not familiar with the targeted markets. Despite a market can be uncertain, and not clear in all its aspects, a good entrepreneur can anticipate and is proactive to unforeseen events [30,31]. Instead of starting gradually with a funnel strategy of how to know the market dynamics and get the first paying customer, the startups wrongly focused on perfecting the business model.

5.2 Neglected Learning Process

As the firms mature and the awareness of the competitive environment grows, there is an increasing reluctance to share ideas, problems or solutions in the wider sense [32]. Indeed, over time learning progress slowed down and the two startups were only concerning how to gather more and more customers. However, improving customer acquisition before problem/solution fit has been premature, because users were not hooked to the product. Involving the customer to activate the learning progress has also been discussed by Yogendra [33] as an important factor to encourage an early alignment of business concerns to technology strategies.

The two startups progressed through their life-cycle inconsistently (see figure 1), addressing specific challenges in the wrong time, such as the need to acquire more and more customers with a growing team, maximizing profits without learning the market dynamics. These strategies prompted further challenges in trying to strike the need of investments, indeed wasting time in further structure and formalization of the business model.

Ultimately, a lost of confidence to achieve independence after using up personal savings, heavily impacted the survival of the companies, as experienced also by McAdam et al. [32]. Learning mechanisms (e.g. learning about one's strengths, weaknesses, skills attitudes etc.) have been widely researched by Cope [34], who reveals a deeper conceptualization of the process of learning from venture failures.

5.3 Implications of the Behavioral Framework

Existing frameworks have recognized the need of aligning business intentions to a set of development activities to achieve a certain goal (e.g. GQM Strategies [35], Balanced Scorecard [36]). However those approaches have been conceived for more established companies, taking in consideration traditional development approaches oriented towards process improvement initiatives.

The described behavioral framework, constructed by means of studies on startup cases, can be used as a tool by practitioners to analyze how consistently they are distributing their focus on the Macmillan et al. dimensions along with the exploration and validation stages. For example, a configuration might appear as figure 2, where the marked strategies (the market and team dimensions) represent a premature scaling towards the validation stage.

As remarked by Blank [2], the product/market fit has the objective of validating people's interest in a proposed product. If it doesn't occur, "pivoting" needs to be applied through changing the product or tackling a problem from a different angle. He continues reporting that if the validation phase is successful, the startup company can start increasing profits and improving their customer acquisition process (also known as scaling phase). However knowing the mismatch of possible dimensions, according to a targeted phase, can prevent the waste of resources and improve the decision process.

In real settings, this framework would allow CEOs to focus and understand how decision-making strategies are aligned to the activities within a software startup company in the early-stage of its development. Moreover, venture capitalists might better understand the stage of development of a startup company and filter those who can simultaneously sustain a consistent focus on different dimensions.

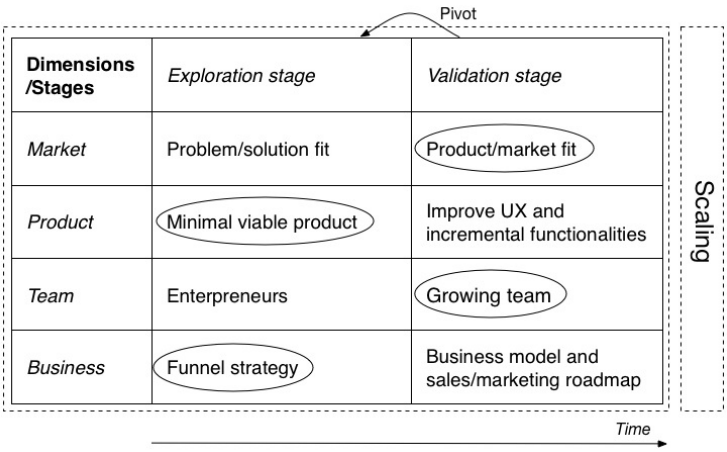

Fig. 2. Behavioral framework: the dimensional mismatch

6 Conclusions and Recommendation for Future Research

Software startups are able to produce cutting-edge software products with a wide impact on the market, significantly contributing to the global economy. Software development, especially in the early-stages, is at the core of the company's daily activities. Despite their severely high failure-rate, the quick proliferation of software startups is not supported by a scientific body of knowledge [5]. This paper provides an initial explanation of failure by means of a multiple-case studies based on two software startups, focusing on early-stage activities, from the *market, product, team* and *business* perspectives.

The behavioral framework is derived from the hindsight knowledge collected from the CEOs of the two failed startups, with the aim of explaining how inconsistent decision-making strategies could lead to failure.

One important validity threat to this study is the small number of cases. However, as described by Klein et al. [37], there is a basis for abstraction and generalization in interpretive field studies through the use of ideas and concepts if rigorously collected and experienced by researchers. As suggested by [38], to validate the explanatory capability and correctness of the model we compared the findings with the state-of-the-art (see section 5). To validate interview data we examined also supporting evidences to verify their expressed opinions, such as emails, presentations and documentation. The two studied startups might also be biased by contextual factors, such as type of product, competitive landscape etc. To mitigate this threat we constructed the framework using Macmillan et al. dimensions, widely used in previous software engineering studies [30,24], enabling a broader reasoning related to the factors that hinder the success of software startups.

The overall results of our study reveal inconsistency between the strategy of understanding and testing the problem/solution fit and the behavioral execution of pursuing the product/market fit. When resources are scarce, survival and success depend most heavily on the executives and managers, who are responsible for shaping, directing, and implementing company strategies. Early recognition and management of critical issues can increase the chances of success for a software startup. The two startups failed to understand the problem and provide the right solution, showing increasing reluctance in learning from the potential customers.

In this paper we integrated a number of novel challenges discovered according the Macmillan et al. dimensions [14] for both practitioners and researchers, while presenting a first set of concepts, terms and activities which set startup strategies for the rapidly increasing startup phenomenon. By means of the behavioral framework, we provided a possible reason for the failure of software startups. Consequently, there is a great deal of scope here for further research. For example, further research is required to investigate the following two outstanding questions; Firstly, how to prevent mismatch between business intentions and development execution? Secondly, how existing learning processes can improve business and development alignment?

References

1. Smagalla, D.: The truth about software startups. MIT Sloan Manage. Rev. (USA) 45(2), 7 (2004)
2. Blank, S.: The four steps to the epiphany, 1st edn. CafePress (February 2005)
3. Crowne, M.: Why software product startups fail and what to do about it. In: Proceedings International Engineering Management Conference (IEMC), pp. 338–343 (2002)
4. Sutton, S.M.: The role of process in software start-up. IEEE Software 17(4), 33–39 (2000)
5. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: A systematic mapping study. Information and Software Technology (forthcoming)
6. Marmer, M., Herrmann, B.L., Dogrultan, E., Berman, R., Eesley, C., Blank, S.: The startup ecosystem report 2012. Technical report, Startup Genome (2012)
7. Coleman, G., O'Connor, R.: Investigating software process in practice: A grounded theory perspective. Journal of Systems and Software 81(5), 772–784 (2008)
8. Christensen, C.M.: The Innovator's Dilemma. Harvard Business School Press (1997)
9. Storey, D.: Entrepreneurship and the New Firm. Croom Helm (1982)
10. Perkins, A.B., Perkins, M.C.: The Internet Bubble: Inside the Overvalued World of High-Tech Stocks – And What You Need to Know to Avoid the Coming Catastrophe. HarperInformation (1999)
11. Marmer, M., Herrmann, B.L., Dogrultan, E., Berman, R., Eesley, C., Blank, S.: Startup Genome Report Extra: Premature Scaling. Technical report, Startup Genome (2011)
12. Ruokolainen, J., Igel, B.: The factors of making the first successful customer reference to leverage the business of start-up software company - multiple case study in thai software industry. Technovation 24(9), 673–681 (2004); Cited By (since 1996): 4
13. Ruokolainen, J.: Gear-up your software start-up company by the first reference customer - nomothetic research study in the thai software industry. Technovation 25(2), 135–144 (2005)
14. Macmillan, I.C., Zemann, L., Subbanarasimha, P.: Criteria distinguishing successful from unsuccessful ventures in the venture screening process. Journal of Business Venturing 2(2), 123–137 (1987)
15. Yu, Y.W., Chang, Y.S., Chen, Y.F., Chu, L.S.: Entrepreneurial success for high-tech start-ups - case study of taiwan high-tech companies, Palermo, Italy, pp. 933–937 (2012)
16. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic Mapping Studies in Software Engineering. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), pp. 1–10 (2007)
17. Yin, R.K.: Case study research: design and methods. Sage Publications (1994)
18. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report (2007)
19. Miles, M., Huberman, A.: Qualitative Data Analysis: An Expanded Sourcebook, 2nd edn. Sage, Thousand Oaks (1994)
20. Kitchenham, B., Dyba, T., Jorgensen, M.: Evidence-based software engineering. In: Proceedings of the 26th International Conference on Software Engineering, ICSE 2004, pp. 273–281 (May 2004)

21. Dyba, T., Kitchenham, B., Jorgensen, M.: Evidence-based software engineering for practitioners. IEEE Software 22(1), 58–65 (2005)
22. Davis, A.: Operational prototyping: a new development approach. IEEE Software 9(5), 70–78 (1992)
23. Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business (2011)
24. Hui, A.: Lean change: Enabling agile transformation through lean startup, kotter and kanban: An experience report, Los Alamitos, CA, USA, pp. 169–174 (2013)
25. Carmel, E.: Time-to-completion in software package startups. In: Proceedings of the System Sciences, pp. 498–507 (1994)
26. Midler, C., Silberzahn, P.: Managing robust development process for high-tech startups through multi-project learning: The case of two European start-ups. International Journal of Project Management 26(5), 479–486 (2008)
27. Hilmola, O.P., Helo, P., Ojala, L.: The value of product development lead time in software startup. System Dynamics Review 19(1), 75–82 (2003)
28. Park, J.S.: Opportunity recognition and product innovation in entrepreneurial hi-tech start-ups: A new perspective and supporting case study. Technovation 25(7), 739–752 (2005)
29. Preston, J.T.: Building success into a high-tech start-up. Industrial Physicist 9(3), 16–18 (2003)
30. Kakati, M.: Success criteria in high-tech new ventures. Technovation 23(5), 447–457 (2003)
31. Hunt, F., Probert, D., Wong, J., Phaal, R.: Valuation of technology: Exploring a practical hybrid model, pp. 47–53 (2003); Cash flow analysis; Product life cycles
32. McAdam, M., McAdam, R.: High tech start-ups in university science park incubators: The relationship between the start-up's lifecycle progression and use of the incubator's resources. Technovation 28(5), 277–290 (2008)
33. Yogendra, S.: Aligning business and technology strategies: a comparison of established and start-up business contexts. In: Proceedings of the Internal Engineering Management Conference (IEMC), pp. 2–7 (2002)
34. Cope, J.: Entrepreneurial learning from failure: An interpretative phenomenological analysis. Journal of Business Venturing 26(6), 604–623 (2011)
35. Basili, V.R., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., Seaman, C., Trendowicz, A.: Linking software development and business strategy through measurement. arXiv preprint arXiv:1311.6224 (2013)
36. Kaplan, R.S., Norton, D.R.: The balanced scorecard: Measures that drive performance. (cover story). Harvard Business Review 83(7/8), 172–180 (2005)
37. Klein, H.K., Myers, M.D.: A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Quarterly, 67–93 (1999)
38. Corbin, J., Strauss, A.: Grounded theory research: Procedures, canons, and evaluative criteria. Qualitative Sociology 13(1), 3–21 (1990)

A Comparative Perspective between Investors and Businesses Regarding Success Factors of E-Ventures at an Early-Stage

Tim Taraba[1], Martin Mikusz[1,2], and Georg Herzwurm[1]

[1] Department VIII, Chair of Information Systems II, Prof. Dr. Georg Herzwurm, Keplerstraße 17, 70174 Stuttgart, Germany
[2] FOM University of Applied Sciences, Rotebühlstr. 121, 70178 Stuttgart, Germany
{taraba,mikusz,herzwurm}@wius.bwi.uni-stuttgart.de

Abstract. Start-ups in the software-intensive field of e-business are key for modern economies. However, those so-called e-ventures tend to face certain problems in terms of financing: Many promising e-ventures seem to fail due to missing seed capital or too few investors. The reasons for this might partially be explained by goal conflicts, different expectations–especially concerning growth of enterprise value and opportunities–, differences in valuation of risks, planning, time horizon and other trade-offs between potential investors and the company's founders. For this reason we examined academic literature to collect data as a basis for two analogously conducted Delphi studies: one for investors and one for e-ventures. Out of 48 most widely researched success factors concerning investors (of technology start-ups) and 24 concerning e-ventures we could derive implications to eight different subtopics for the above-mentioned trade-offs. Our article concludes by naming its major limitations as well as future research directions for the purpose of advancing research in this field.

Keywords: Electronic-Entrepreneurship, IT-Entrepreneurship, Electronic-Ventures, IT-Ventures, Delphi Study, Software Business, Software Startups, Software Companies, Future Software Business, Success Factors.

1 Introduction

Business foundations stimulate the economic dynamics a working national economy. Start-ups in the software-intensive field of e-business; including e-commerce, mobile business, mobile game and entertainment software, hereafter called e-ventures, are crucial in this environment. Facing the increasing penetration of the internet (internet of things and services), many new business models arise. E-business and its sub-sectors hereby belong to the most dynamic and innovative economy sectors. Compared to the overall economy and the remaining high-technology sector, it can be stated that a consequently and significant higher founding dynamic exists in the above-said economic field. However, many of these newly founded e-ventures fail right at their beginning, during their so-called early-stages.

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 42–57, 2014.

According to a study of the Centre for European Economic Research [1], thin capitalization at the beginning of business activity is a major reason for the failure of those ICT-start-ups. Approximately two-thirds of all failed ICT-start-ups attribute their failure to missing seed capital or no, resp. too few investors [2]. We assume that a noteworthy part of these failures is neither based on lacking sustainability of business models nor on crude business plans. Rather, our assumption, particularly regarding e-ventures, has been as follows: the reasons why potentially successful business ideas of e-ventures have not been funded are e.g. goal conflicts, different expectations, especially concerning growth of enterprise value and opportunities, differences in valuation of risks, planning and time horizon as well as other possible tensions between investors and the company founders during the early-stages of e-business venturing.

These reasons form the following problem statement: founders of e-ventures operationalize the term of success unlike investors and in line with this act on different assumptions of critical success factors at planning their business resp. evaluating a business plan. One thinks here of some today´s big players in e-business like eBay, Google or PayPal. From where we stand now, they all have to be considered as successful (or former) e-ventures, which have had the same single problem to deal with: rejected financing by investors during their early stages.

Previous research efforts have been primarily investigating either success factors of start-ups (e-ventures, ICT-start-ups or start-ups in general) [2] or decision processes and rating criteria of investors [4, 5, 6, 7, 8, 9, 10, 11]. We tried to integrate the findings from both perspectives up to now. We have therefore jointly conducted two Delphi studies, both based on results of empirical secondary research. In this context, this type of study can be explained as a special form of a repeated, structured and written survey, which uses experts as respondents. It is, in general, an adequate method for forecasting and evaluating situations which are characterized by incomplete and uncertain knowledge [12]. Our first Delphi study, from a founder's point of view, has identified critical success factors, which e-ventures seem to take as a starting point for their future business planning during their early-stages. Our second study on the other hand, from an investor's perspective, has pursued the similar goal whereby the rating criteria of investors implicitly represent their assumptions concerning critical success factors of e-ventures during their early-stage in nearby future.

This paper therefore focuses on contrasting the results obtained from both studies. It offers valuables information to the high failure rates of e-ventures' funding efforts during the early-stages and implies possible fields of action to increase their chances in terms of successful funding.

The paper is structured as follows: first, the methodology and data collection are explained. Then the descriptive results of the research are shown, including the results for both our studies as well as related implications concerning several subtopics. Our article concludes with explanation of its most important limitations and possible future areas of research.

2 Methodology and Data Collection

2.1 Delphi Method

The Delphi method is a systematic survey method, with which experts on specific issues are interviewed in two or more rounds [12]. The panel receives aggregated and anonymous evaluation feedback in form of univariate statistics (mean, standard deviation, quantiles, interquartile range) after each round [12]. In light of these information, thinking processes are to be triggered and, thus, the participants are encouraged to reconsider their earlier answers [13]. Key characteristic of the method is therefore to focus and build a consensus among all participants by supporting group communication. This is accomplished indirectly by the above-said anonymous feedback, which thereby leads to an increased consensus among the participants and which, at the same time, avoids undesirable opinion leadership effects [12, 14].

We chose the Delphi method especially with regard to counteracting the lack of predictive power of research on past-oriented success factors. Hence, we linked retrospective knowledge from research in the field with prospective information, obtained by the Delphi studies. In our view, the determinants of success for organizations necessarily have to be based on prospective input, i.e. input which does not rely exclusively on retrospective statements, in order to gain deeper insight into the highly dynamic environment of e-ventures during their early-stages.

The common approach of a Delphi study can be divided into the following four steps [12, 14, 15]:

1. Operationalization of the question. Based on the underlying question, items must be established, which are then submitted to the panel to be evaluated. These generated items can either be pre-defined or collected through a first qualitative survey round.
2. Elaboration of a standardized questionnaire and survey. The established items are processed to be quantitatively evaluated by the expert panel. This is done with the help of a fully or partially standardized questionnaire.
3. Provision of feedback on the previous round. Standards on how this feedback can be given hitherto hardly existent. In short, most of the time descriptive statistics are used here, above all robust measures of dispersion and location. The aim is to illustrate the given diversity of opinions in a simply and readily understandable way.
4. Repetition of the survey. Reflection as well as reconsideration of earlier answers given by the experts–based on the feedback.
   If necessary, this step is repeated multiple times.

2.2 Design of the Conducted Studies

Due to the broad research basis for success factors of start-ups as well as on investor´s rating criteria, we have pre-defined items in both our studies, instead of collecting them through e.g. a first qualitative survey. Accordingly, we have elaborated our standardized questionnaires based on empirical secondary research.

The questionnaires were developed using an HTML-based survey software. Hence, the received it via e-mail and answered it online.

Both studies were conducted in two rounds. According to appropriate literature, a satisfying result is achievable within a maximum of only three rounds whereas the most striking changes usually arise between the first two rounds of a Delphi study [12, 15]. As for our study, the conducted consensus analysis gave no evidence leading to the necessity of a third round. For the purpose of examining the general process of building a consensus among the experts we have calculated the coefficient of variation for each item of both rounds and studies. Our results indicate that we achieved satisfying results for both studies in regards to the improvement of judgment, resp. reaching consensus.

The same questions were asked in both rounds. To do so we placed aggregated, anonymous feedback on each item of the first round directly in each of the second round's questionnaires. Concerning this matter, we decided to use box-whisker-plots, which included mean, quantiles and interquartile range for each item. Box-whisker-plots outline various robust measures of location and dispersion in an understandable and convenient way in just one graphic. To avoid misinterpretations, we also added information to the questionnaire which illustrated the used form of box-whistler-plots and explained, how they were to be interpreted in order to reassess every item.

In order to ensure the absence of certain kinds of errors, regarding question content and formatting concerns which are often associated with survey research, we have pretested both questionnaires for both rounds in pre-field and field. Resulting improvements mainly applied to correct presentation and comprehensibility of the box-and-whisker plots in all major internet browsers (Google Chrome, MS Internet Explorer, Apple Safari and Mozilla Firefox). Multiple adjustments to the CSS-Style sheets of the online questionnaire had therefore to be made.

The total duration of our two survey periods were nine weeks. This applies to both rounds. The studies took place between March 2013 and May 2013.

2.3 Delphi Study of the Investor´s Perspective

To identify the main rating criteria in regards of potential investors' decision-making-processes, we conducted a comprehensive review of available secondary literature. Premier research databases including EBSCO, Science Direct, Google Scholar and ACM were searched using a preset of defined keywords. To ensure high quality of used literature, we only took international peer-reviewed journal publications into account and examined the suitability and content validity of each study. Only empirical studies that included the evaluation process of ICT-start-ups during their early-stages from an investor's point of view were used. In addition, crucial citations, referenced within these set of publications, were examined and–if applicable–included. The studies mostly covered a certain groups of investors (venture capital firms, business angels, incubators, et cetera). To serve as a basis for the work at hand, these criteria were cataloged. This process ultimately lead to an intersection of all investor groups, hence, equally relevant for all types.

In the end, 47 criteria from 13 studies remained to be operationalized [see Table 1]. In the first instance we attempted to draw on items that had already been used in one of the above-mentioned surveys. If, however, this was not possible because or especially since the former questionnaires have not been published in detail, new items were formulated by us. Since the length of our questionnaires was a major constraint, we operationalized each criterion by not more than two items. Finally, 58 items related to the rating criteria and 5 required items regarding meta-information about the participating investors were included in the questionnaire.

These 58 former mentioned items were enquired in 5 equally formatted item groups (13 on founder´s, 17 on product/service and 12 on market characteristics, and 8 per financial and investment perspective). The guiding question remained the same: "What influence are the following statements going to have on investment decisions you have to asses in the coming three years?" A standardized unipolar 6-points-scale was applied for judging the influence. This so called "forced choice" method was used to avoid neutral answers. It was, however, possible to select an option which said "I don't know" to prevent a bias resulting out of forced positive/negative estimations.

2.4 Delphi Study of the Founder´s Perspective

The current state of research in the addressed field allowed us to abstain from conducting our own comprehensive reviews of secondary literature. Instead, we used a meta-analysis as a foundation published by Song et al. in 2008. Using Pearson correlations as effect size statistics, Song et al. analyzed the findings of 31 selected primary statistical analyses from research of success factors concerning new technology ventures (including e-ventures) and, in doing so, identified the 24 most widely researched. Further performed analysis showed that, among these, 8 factors were homogeneous significant and 11 remained heterogeneous significant. For the heterogeneous success factors, an additional moderator analysis was conducted [3].

Detached from the significances, homogeneities and further implications which derived from Song et al., we took the initial identified 24 success factors as a basic starting point for our own operationalization. We did this to satisfy the explorative orientation of our study and–out of the educated guess–that success factors of e-ventures, as partial quantity of young technology ventures, could perhaps underlie deviant causalities. Where possible, we used the same items which as in the empirical studies examined by Song et al.

Beside two exceptions, we operationalized the success factors by use of two items per factor. All of our 45 items were prompted in 6 equally formatted item groups (6 per market and product, 8 per strategy, enterprise and resources, 9 regarding management). The items resp. success factors had to be evaluated in terms of effectiveness as well as controllability for the single e-venture. To avoid misinterpretations, a short explanation of both expressions were provided: effectiveness describes how strong a factor influences the success of an e-venture regarding the next 3 years. Controllability expresses to what extend an e-venture is able to change, secure or induce this factor to its own use. For judging effectiveness and controllability a standardized unipolar 6-points-scale was applied (see chapter 2.3).

Besides of the above factors, the participants were questioned a variety of questions concerning meta-information regarding person, role, enterprise and–to ensure comparability of the empiric data–manner of success measurement in the actual company (e.g. revenue, profit). Eventually, the participants had the opportunity to bring in further items resp. success factors which went beyond the given ones. They were also able to judge effectiveness and controllability of these new factors. However, no significant outcome resulted from this and hence, to build upon the meta-analysis of Song et al. turned out to be an expedient approach.

3 Descriptive Statistics

3.1 Delphi Study of the Investor´s Perspective

Our target population has consisted of investment managers of venture capital firms, incubators and business angels. The panel had to assess start-ups, more specifically e-ventures, during their early stage phase. Turning to the response rate, a total of 209 potential respondents clicked on the hyperlink, which lead to the first round of our survey. Out of these 209 respondents we could attract 102 to see the second side of the questionnaire and a total of 66 completed it entirely. The second survey round had a total of 66 respondents of which 63 saw the second page. All in all we could achieve 51 full questionnaire replies for the second round.

All of our interviewed participants have been dealing with e-ventures; two-thirds have placed more than 50% of their investments in e-ventures. In average, the participants have had 8.5 years of work experience in the field of new venture financing; 82% have had assessed more than 100 business plans. Taking everything into consideration, the size and quality of the panel can be safely regarded as adequate for conducting a Delphi survey. Furthermore, the calculated coefficient of variation indicated, that the achieved consensus generated through the Delphi process had improved for all items, except one–which was market growth. It is therefore important to bear in mind the possible bias of this particular item.

The following Table 1 shows the basic results of our assessment.

Table 1. Factors considered to affect investor's future decision-making process

Factor	Code	Explanation	I.[a]
customer value	I_1	p/s offers a noticeable additional benefit	5,87
reliability	I_2	honest and earnest behavior	5,68
industry exp.	I_3	knowledge regarding the branches/industries the business idea addresses	5,30
market volume	I_4	sales targets can be reached in the targeted market	5,30
available market	I_5	demand for p/s is existent	5,28
degree of innovation	I_6	p/s is an essential improvement of an existing solution	5,25
financing phase	I_7	start-up finds itself in a phase that is usually supported by you / your institution	5,23
know-how	I_8	commercial and technical knowledge existent	5,19

Table 1. (*continued.*)

exp. of the team	I_9	complementary knowledge among team members	5,19
market growth	I_10	target market shows a high level of growth	5,19
growth potential	I_11	target market is currently a niche market, but shows growth potential	5,04
real. forecasts	I_12	generally, realistic forecasts are made	5,02
prototype	I_13	a prototype of the product is existent	4,98
capital demand	I_14	capital demand to reach the break-even point	4,96
market entry barriers	I_15	possible existence of market entry barriers	4,83
market develop.	I_16	high amount of early-adopters in targeted market segment	3,96
	I_17	realistic penetration strategy for the market	5,64
previous results	I_18	fast realization of the business idea up to now	4,81
copyrights	I_19	start-up owns copyright on p/s	4,81
product type	I_20	type of product (hardware, software, service)	4,68
beta-tests	I_21	beta-tests show huge interest	4,64
imitability	I_22	p/s can potentially be protected by a copyright	4,36
	I_23	p/s can only be copied by investing high amounts of time and money	4,85
commitment	I_24	quit old job to fully focus on the business idea	4,13
	I_25	willingness to invest one's own capital	5,02
lump sums on progress	I_26	capital payment can be based on progress in the project	4,64
expected ROI	I_27	projected ROI equals those of other start-ups supported by me or my institution	4,40
competition intensity	I_28	target market is a new market segment without direct competitors	4,66
	I_29	target market is very fragmented with a high number of potential competitors	4,06
	I_30	few but dominant competitors are active in the target market	4,30
distributorship	I_31	possibility to enforce distribution partnerships	4,34
develop. progress	I_32	p/s will reach marketability within the next 6-12 months	4,94
	I_33	p/s is in an early stage of development	3,64
invest. period	I_34	estimated investment period	4,28
prof. experience	I_35	have several years of professional experience	4,23
exit-options	I_36	business plan includes exit-opportunity	4,26
	I_37	type of planned exit	4,08
market character	I_38	target market is a niche market	4,02
	I_39	target market is a mass market	4,15
unique selling point	I_40	p/s is an innovation	5,09
	I_41	p/s is a copy of a functional idea (e.g. from the USA)	2,94
executive ability	I_42	already assumed management responsibility in other projects	4,00
develop. risk	I_43	is bases on the development knowledge of a single person	3,53
	I_44	p/s uses new, rarely used technologies	3,96

Table 1. (*continued.*)

active investors	I_45	co-investors are active partners	3,70
travel time	I_46	travel time between you/your institution and the potential investment	3,66
profitability	I_47	break-even point is reachable without further financing rounds	3,64
other investors	I_48	there are already other investors involved in the start-up	3,53
portfolio-fit	I_49	start-up in own portfolio works on a similar business idea	3,53
past achieve.	I_50	have participated in prior venture creation	3,51
recommendation	I_51	recommendations of a different investor	3,70
\references	I_52	recommendations of a different entrepreneur	3,04
exp. as a team	I_53	team-members already worked together in the past	3,21
supply-chain	I_54	complex supply chain (for hardware products)	3,04
inactive invest.	I_55	co-investors are inactive partners	2,81
cash-flow	I_56	up to now no profit was generated	2,28
situation at financial markets	I_57	tense situation on the international financial markets	1,96
crowdfunding	I_58	first round of funding via crowdfunding successful	1,58

a influence

3.2 Delphi Study of the Founder´s Perspective

Our panel on the entrepreneurial side of view has consisted of both, founders and managerial directors of e-ventures during their early-stage. A total of 337 respondents clicked on the hyperlink which lead to the questionnaire of round one. 157 respondents viewed the welcome page (first page) and 53 completely answered all questions. On the whole we could collect 53 complete questionnaires in round one. As to the second round we could manage all 53 respondents to click on the hyperlink leading to questionnaire number two. Finally, a total of 33 evaluable questionnaires from members of the target group (which was ensured with the help of corresponding requested meta-information) were submitted to us after the end of the second round of the survey. Despite of the slightly lower participation in comparison with the investor´s perspective, the size of the panel and its quality can still be safely regarded as adequate. In this respect it is noted that experiments verified: differences in results of a 16-participant panel are less than 10% in comparison to a 32-participant panel [21].

The majority of participating e-ventures have been companies (mostly Ltd.) at the beginning of product production or market launch. All of our interviewed e-ventures found themselves in the so-called early-stages. The company size varied between two groups, one had mostly below 50 employees and one between 50 and 250. The companies' average age varied between 1-2, resp. 3-4 years. 10 of 33 participating e-ventures have been offering software; 26 (software-intensive) services and 5 hardware. Thus, multiple belongings to the last category were possible. As to the generated consensus among our participants, the calculated coefficient of variation

indicated improvement for 40 out of 45 items in terms of the effectiveness. Concerning the Controllability, even 44 of 45 items showed improvement.

Table 2 summarises our basic findings.

Table 2. Factors considered to affect e-ventures future business success

Factor	Code	Explanation	E.[a]	C.[b]
marketing intensity	E_1	unique product image	5,03	4,97
	E_2	unique firm image	4,94	5,15
marketing experience	E_3	marketing experience of the management	4,73	4,76
	E_4	marketing know-how of the management	4,88	4,94
industry experience	E_5	branch experience of the management	4,69	4,52
	E_6	market knowledge of the management	4,82	4,76
product innovation	E_7	focusing on R&D	3,12	4,61
	E_8	offered product varieties/differentiations	3,94	5,15
market scope	E_9	heterogeneous customers/customer segments	3,82	3,79
	E_10	broad product portfolio	3,70	5,45
R&D alliances	E_11	horizontal R&D cooperation	3,37	4,07
	E_12	strategic alliances	4,61	4,47
prior start-up experience	E_13	entrepreneurial experience of the management	4,00	4,45
	E_14	management experience in similar positions	4,00	3,82
environmental heterogeneity	E_15	broad operating domain	3,68	3,87
	E_16	branch-specific business processes	3,87	4,65
financial resources	E_17	high capital base	4,36	3,79
	E_18	high equity ratio	3,88	4,03
internationalization	E_19	international experience of the management	3,66	3,86
	E_20	foreign business transactions	3,69	4,66
market growth rate	E_21	increasing total market volume	5,33	3,39
	E_22	increasing demand	4,76	2,24
R&D experience	E_23	R&D experience of the management	3,68	4,00
	E_24	R&D know how of the management	3,65	4,29
nongovernment. financial support	E_25	investor financing	3,94	3,73
	E_26	other private funds	3,73	3,76
supply chain integration	E_27	product development with suppliers	3,21	3,75
	E_28	intensive communication with suppliers	3,53	4,38
firm type	E_29	managerial independence	4,58	4,48
	E_30	support from possible parent company	3,03	2,53
firm age	E_31	long company existence	3,24	2,42
firm size	E_32	amount of permanent employees	3,33	4,16
	E_33	amount of free collaborators (freelancer)	2,79	4,19
low cost strategy	E_34	low sales prices	3,22	4,06
	E_35	low purchase prices	3,70	2,70
university partnerships	E_36	university research cooperation	2,91	3,88
environmental dynamism	E_37	constant consumer preferences	4,03	2,48
	E_38	fast technological change	3,91	2,85
competition intensity	E_39	absence of substitute products	4,19	1,88
	E_40	low intensity of competition	4,61	2,06
size of founding team	E_41	amount of foundation members	2,52	3,25
R&D investments	E_42	increased budget for patent application	1,20	2,66
	E_43	recruitment of R&D-staff	2,52	3,24
patent protection	E_44	ownership of important patent rights	1,87	2,73
	E_45	increasing patent application	1,47	3,14

a effectiveness, b controllability

4 Analysis and Implications

Taking the diverged results of both our Delphi studies into account, a variety of implications arise, which are to be discussed in what follows. By examining our findings, we suggest to separate the following eight subtopics: Experience and know-how, financial situation, copyrights and imitability, demand, market size and market volume, supply situation and competitive environment, innovation, value added and image, agency and portfolio aspects, strategic alliances and partnerships.

4.1 Experience and Know-How

Taking into account our results concerning the entrepreneurial team's experience and know-how, they point towards evidence that entrepreneurs themselves tend to overestimate entrepreneurial experience of their management team (E_13, E_14) in comparison to the estimations we received by asking potential investors about the importance of having participated in prior venture creation (I_50). Investors therefore seem to consider the said factor to be less–but still–crucial. If it comes to managerial experience, our results implicate that both parties regard it as an important factor (E_14, I_42). Especially knowledge in conjunction with the target market as well as the industry appears to be essential for both investors and entrepreneur (E_5, E_6, I_3). There is also indication that entrepreneurs emphasize individual competencies such as R&D experience of the management (E_23), R&D know-how of the management (E_24), international experience of the management (E_19), marketing experience of the management (E_3) and marketing know-how of the management (E_4) whereas investors potentially tend to focus on technical expertise and commercial knowledge of the approached e-venture (I_8, I_35). It can be seen from the above data that particularly marketing seems to be of importance for the e-venture's future success. Another important finding was that existence of complementary knowledge amongst team members may be suggested as a major determinant regarding the potential investor's decision-making processes (I_9). In summary our results suggest that entrepreneurs emphasize marketing know-how and experience together with industry and market knowledge. Neither international experience nor entrepreneurial and managerial skills or experience seem to be crucial. Likewise our study suggests that R&D experience and know-how are rather unimportant for e-venture's future success. In contrast potential investors seem to consider complementary competences of the e-venture's team as vital. Our results also indicate that they focus on commercial skills and overall expertise in almost the same manner as e-ventures but set no great story by know-how and experience in prior entrepreneurial activities.

4.2 Financial Situation

As can be seen from our results, especially successful financing through crowdfunding has to be considered as a relatively irrelevant factor for investors (I_58). In addition the results, as shown in Table 1, indicate that participation of other

investors whether and in what form can be suggested as relatively insignificant as well (I_45, I_55). Especially if further investors belong to the category of so called silent partners, they seem to not play a big role (I_55). An explanation might be that their existence possibly has no effect regarding the number of direct stakeholders of the particular e-venture. Existence of active partners on the other hand can be suggested as a more relevant decision criterion since they happen to erode e.g. the former single investor's opportunistic control (I_45). With attention to the entrepreneurs it can be suggested that financing via investors is a rather crucial success factor (E_25). Our results also indicate that private funding is believed by entrepreneurs to be another major pillar (E26). It seems possible that these results are due to easy acquisition of private funds in comparison to commercial investments. Private funds moreover maintain entrepreneurial freedom and are therefore to be welcomed. There are several possible explanations for this. One might be that private funding is not as opposed to the typical interest in profits which commercial investors tend to drive at. In summary the answers given by entrepreneurs indicate that–as we expected–the financial situation must be considered a fairly important success factors which also seems to be pretty influenceable for the particular e-venture (E_25, E_26).

4.3 Copyrights and Imitability

It is apparent from both our tables that copyrights and general imitability perceived to be of little relevance for e-ventures in terms of future business success whereas a great number of potential investors we interviewed regard them as pretty important (I_19, I_22, I_23, E_42, E_44, E_45). It can also be suggested that entrepreneurs seem to be capable to influence those factors to a certain extent (E_42, E_44, E_45). Investors on the other hand appear to attach great importance to ownership of copyrights as well as the future patentability of potential products and/or processes (I_19, I_22). This rather contradictory result may be due to the ventures limited resources in the sense of capital and time. This leads to the assumption that copyrights and imitability are perhaps not mandatory in comparison to more important factors such as e.g. marketing. It may also be the case that investors on the other hand consider e.g. copyrights as discrete values of a company which–in case of a discontinuation of business or insolvency–remain existent and therefore leave the opportunity to be converted into cash. There are, however, several other possible explanations which this study has been unable to demonstrate.

4.4 Demand, Market Size and Market Volume

The demand for products and services seem to be quite important for e-ventures as Table 2 shows (E_21, E_22). As assumed, especially the demand for products on offer (E_22) and the general market volume (E_21) appear to be crucial. Interestingly, most of the potential investors we surveyed also considered market development to be a relevant factor beside of the above mentioned (I_4, I_14, I_16, I_17, I_21). They, in this regard, appear to put attention to e.g. presence of a realistic market penetration strategy (I_14, I_17). It can therefore be assumed that e-ventures, since their business

plan during the early-stages is already set up, tend to focus on achieving their market objective whereas investors on the other hand seem to first evaluate the business concept of the e-venture, including the latter determinants such as market entry barriers.

4.5 Supply Situation and Competitive Environment

Our results as regards supply situation and competitive environment (I_15, I_28, I_29, I_30, E_39, E_40) imply that they are both of slightly limited relevance in comparison to the factors mentioned in chapter 4.4. Thus, they seem to be still of great importance for both–inventors and e-ventures. There is evidence that entrepreneurs consider competitive intensity to be very important (E_40). Investors, in contrast, rather seem to focus on market entry barriers (I_15). This is in itself somewhat incomprehensible due to the fact that detailed information about the supply- and competitive situation is required to give meaning to two of three aspects we examined in the previous subchapter, namely: market volume and market penetration strategy. Our results possibly result due to a different level of abstraction which both categories–investors versus entrepreneurs–of studies we examined faced. They therefore have to be interpreted with caution as further studies will need to be undertaken. In the final analysis the same explanation as for subchapter 4.4 may be applied to explain these findings. This means that an existent product may lead to a certain market which then eventually results in confronting a given competitive environment. Taken together, it hence seems that factors such as market barriers or existence of competitors are already determined for e-ventures during the early-stage. An implication of this is the possibility that those factors are less important since ambitions of influencing them may not be rewarded. Investors on the other hand seem to find themselves in a position where the general decision of whether or not to invest is the one to be made, which is why they perhaps appear to face an extended scope and therefore tend to also concentrate on actual market characteristics.

4.6 Innovation, Value Added and Image

It is apparent from Table 1 that investors seem to assume a demand for the product to be especially driven through the level of innovation (I_6, I_40) and the value added for customers (I_1). Thus, both criteria can be considered to constitute two major success factors concerning the investor's decision making process. From the data we can also see that it seems to be relevant whether the offered product is a material good, a service or a software (I_20) as well as if e.g. a prototype exists (I_13). It is–on the other hand–apparently not very important for investors whether there e-venture which is potentially to be supported builds on new and/or rarely used technologies (I_44), is bases on the development knowledge of a single person (I_43) or is a copy of a functional idea, e.g. from the USA (I_41). Entrepreneurs in contrast seem to singly and solely emphasize the image; to be precise: product and firm image (E_1, E_2). Aside of that they also appear to regard the offering of product variations and –differentiation to be crucial for success and easy to influence (E_8). This discrepancy

is interesting to note and may be explained by the assumption that e-ventures are prone to achieve the best possible improvement of a given product, e.g. via intense customer interaction since their initial product/service is already designated. As a result many choices a potential investors may has, are inapplicable for the e-venture by itself. In summary it can be stated that investors appear to consider soberly the customer's benefit compared to other existing solutions which are already available at the market. Entrepreneurs, as mentioned before, seem to mainly focus on product and firm image.

4.7 Agency and Portfolio Aspects

As Table 1 shows, there is evidence that potential investors consider multiple factors as relatively crucial which can be grouped under the category of so-called agency aspects (I_2, I_12, I_18, I_24, I_25, I_26, I_36, I_46, I_47, I_51, I_52). Those factors, such as recommendations of a different investor or entrepreneur as well as the willingness to invest one's own capital–concerning the particular e-venture which is to be financed–, seem to be important since they may provide indication as to the actual motivation and impetus of the respective entrepreneur. The current study found that entrepreneurs, on the other hand, appear to have no direct equivalent to the above factors. The reason for this is not clear but it may have something to do with the axiomatic information asymmetry arising from the principal-agent-relationship which the investor (principal) and the entrepreneur (agent) are subject to. Hence, the agency aspects are important for the former because it may help him to explore the entrepreneur's real intent and future behavior. It may be that e.g. recommendations of a different investor/entrepreneur and honest or earnest behavior are especially relevant prior to the formation of a contract. As said, this may also be due to the fact that some of these factors grant indication towards the entrepreneur's real objectives (I_24, I_25). If, besides of that, capital payment are based on progress in the project this may also lead to goal congruence via harmonizing the incentives of both parties (I_26). E-Ventures on the other hand seem to be able to use so-called signaling to convince the potential investors. In this context they could e.g. attach information about exit-options (I_36, I_37), realistic forecasts (I_12) or evidence on rapid progress (I_18) to their written business plan.

If it comes to portfolio aspects, our results unsurprisingly suggest that, again, they are quite relevant for investors (I_7, I_27, I_37, I_49) but not as much for entrepreneurs, as no counterparts could be identified by our research. These findings may be explained by the fact, that most capital-seeking e-ventures cannot chose between a varieties of possible investors, but rather count themselves lucky when finding just one to support them. It may also be the case that ventures–in comparison to investors–are generally better informed about (expected) actions of other party. This could make further examination superfluous.

4.8 Strategic Alliances and Partnerships

From the data in Table 1 and 2, it is apparent that the existence of strategic alliances and partnerships is rather important for both the entrepreneurs and the investors (I_31,

E_12). There was, however, no equivalent on the same level of abstraction for investors, even though the possibility of distribution partnerships can be named a decision criterion (I_31). An implication of this is the possibility that e-ventures during the early-stages are able to underpin information-based products or services by establishing strategic alliances and partnerships. Investors in contrast possibly consider sales targets as more crucial. There are, however, other possible explanations this study cannot provide. It also remains questionable whether or not the existence of the above-named alliances and partnerships have to be seen as crucially important for investors.

5 Limitations

Most studies concerning research on the factors of success are generally to be questioned in opinion of some authors such as e.g. March and Sutton [16]. We, however, feel that most of this critique is inapplicable for both of our studies. We chose the Delphi method particularly with regard to counteracting the lack of predictive power of past-oriented success factors research. As explained, we linked retrospective knowledge from success factor research with prospective information, obtained by our Delphi studies. Furthermore, we did not propose any normative statements regarding corporate success and reasons leading to it, but we do aim towards an exploration of the contradictive assumptions which founders on the one hand and investors on the other, underlie. It must be stated, that the Delphi procedure itself is primarily criticized because of reasons carried out by H. Sackmann [17, 18]. Although fundamental doubts are–according to today's state-of-the-art scientific research [12, 19, 20]–widely eliminated, this methodological critique must also be applied to our work. It regards e.g. the general effectiveness of the given feedback. Also, operative limitations must be taken into account. For instance, we assume that neither the list of underlying success factors nor that of rating criteria is complete. Another problem might arise due to the fact that the quality of this work is in some parts dependent on the accuracy of the used secondary literature, whereby we best possibly checked their quality and therefore assume it as sufficient. Beyond that, the gained data can by no means be seen as representative and should rather be consulted as first reference points for further, more extensive studies which approach a superior research panel.

6 Conclusions and Future Research Directions

Taken together, this study has gone some way towards enhancing the understanding of the relationship between e-ventures and potential investors in regards to future success factors. Our Delphi study, concerning the founder’s point of view, has identified critical success factors, which e-ventures are going to take as a starting point for their business planning for the coming three years. Our second study, from the investor’s perspective, has pursued the similar goal, whereby the rating criteria of investors implicitly represent their assumptions about critical success factors of

e-ventures. Thus, this paper focused on contrasting the results obtained from both studies resp. point of views, it remains to be emphasized that the link between classic, past-oriented success factor research and forecasts gained by the Delphi process provide a new contribution to this particular field of research. Through comparing both points of view–e-ventures versus investors–it can therefore be suggested that our findings possibly serve as a base to understand future decision-making processes to be made by both the parties we examined. It is, however, up to future research to take up this first exploration of the topic. With respect to the above-mentioned limitations, it is apparent that a number of possible follow-up studies, which use the same mythological setup, are necessary to e.g. verify deducible hypothesizes and/or causal relations.

References

1. Falk, U., Heger, D., Metzger, G., Höwer, D.: ZEW: Ursachen für das Scheitern junger Unternehmen in den ersten fünf Jahren ihres Bestehens, Study for The Centre for European Economic Research, Mannheim (2010)
2. BITKOM Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.: Die Finanzierungssituation von ITK-Start-ups - Hauptergebnisse einer empirischen Analyse des BITKOM, http://www.bitkom.org/files/documents/Finanzierungssituation_ITK-Start-ups(1).pdf
3. Song, M., Podoynitsyna, K., Van Der Bij, H., Halman, J.I.: Success Factors in New Ventures – A Meta-analysis. Journal of Product Innovation Management 25, 7–27 (2008)
4. Macmillan, I.C., Siegel, R., Narasimha, P.N.S.: Criteria used by Venture Capitalists to evaluate new Venture Proposals. Journal of Business Venturing 1, 119–128 (1985)
5. Siskos, J., Zopounidis, C.: The Evaluation Criteria of the Venture Capital Investment Activity: An interactive Assessment. European Journal of Operational Research 31, 304–313 (1987)
6. Hofer, C.W., Hall, J.: Venture Capitalists' Decision Criteria in New Venture Evaluation. Journal of Business Venturing 8, 25–42 (1993)
7. Kakati, M.: Success Criteria in high-tech New Ventures. Technovation 23, 447–457 (2003)
8. Kaplan, S.N., Strömberg, P.: Characteristics, Contracts and Actions: Evidence from Venture Capitalist Analyses. Journal of Finance 59, 2177–2210 (2004)
9. Baum, R.J., Khanin, D., Mahto, R.V., Heller, C.: Venture Capitalists' Investment Criteria: 40 Years of Research. Small Business Institute Research Review 35, 187–192 (2008)
10. Franke, N., Henkel, J., Gruber, M., Harhoff, D.: Venture Capitalists' Evaluations of Start-Up Teams: Trade-Offs, Knock-Out Criteria, and the Impact of VC Experience. Entrepreneurship Theory and Practice 32, 459–483 (2008)
11. Petty, J.S., Gruber, M.: "In Pursuit of the real Deal" - A longitudinal Study of VC Decision Making. Journal of Business Venturing 26, 172–188 (2011)
12. Häder, M.: Delphi-Befragungen – Ein Arbeitsbuch, 2nd edn., Wiesbaden (2009)
13. Becker, D.: Analyse der Delphi-Methode und Ansätze zur ihrer optimalen Gestaltung, Zürich (1974)
14. Linstone, H.A., Turoff, M.: The Delphi Method – Techniques and Applications, London (1975)
15. Woudenberg, F.: An Evaluation of Delphi. Technological Forecasting and Social Change 40, 131–150 (1991)

16. March, J.G., Sutton, R.I.: Organizational Performance as a Dependent. Organization Science 8(6), 698–706 (1996)
17. Sackman, H.: Delphi Assessment – Expert Opinion, Forecasting, and Group Process, Santa Monica (1974)
18. Sackman, H.: A Delphi Critique – Expert Opinion, Forecasting, and Group Process, Massachusetts (1975)
19. Häder, M., Häder, S.: Neuere Entwicklungen bei der Delphi-Methode – Literaturbericht II. In: ZUMA-Arbeitsberichte, Mannheim (1998)
20. Kuhn, J.: Kommerzielle Nutzung mobiler Anwendungen – Ergebnisse der Delphi-Studie "Mobile Business", Regensburg (2004)
21. Duffield, C.: The Delphi Technique – A Comparison of Results Obtained Using two Expert Panels. International Journal of Nursing Studies 30(3), 227–237 (1993)

From Agile Software Development to Mercury Business

Janne Järvinen[1], Tua Huomo[2], Tommi Mikkonen[3], and Pasi Tyrväinen[4]

[1] F-Secure, Helsinki, Finland
janne.jarvinen@f-secure.com
[2] VTT, Oulu, Finland
tua.huomo@vtt.fi
[3] Tampere University of Technology, Tampere, Finland
tommi.mikkonen@tut.fi
[4] University of Jyväskylä, Jyväskylä, Finland
pasi.tyrvainen@jyu.fi

Abstract. The rapid downfall of the Nokia software ecosystem has radically altered the landscape of software industry in Finland in recent years. There has been a shift from largely corporate driven way of working, which is often dominant in large companies, to more agile practices, and in general software organizations are seeking new, leaner ways of composing, delivering, and using software also inside already established companies. To accelerate this transformation in large scale, a collaborative research program has been created, called Need for Speed (N4S). In this paper, we give an insight to the joint goals and concrete actions of the program and discuss the motivations of individual companies that are participating in the program. As one concrete goal of the project, we introduce the concept of Mercury business, where the principles of the Lean startup framework are applied in a more conventional industrial setting.

Keywords: Real-time value delivery, deep customer insight, lean startup, elastic enterprise, mercury business.

1 Introduction

The rapid downfall of the Nokia software ecosystem has radically altered the landscape of software industry in Finland in recent years. Instead of a single ecosystem that has been aiming at the creation of software – including hardware elements, low-level software, operating systems, middleware, and applications – for mobile phones, where major up-front R&D investment has been the norm, smaller companies as well as startups are now becoming major actors. Many of the companies are only working with one layer of the software stack, and for those that work with end-user applications, it is has become crucial to deliver new features to end users whenever they are ready, not when the whole software stack requires updating. This change has meant that new ways of composing, delivering, and using software are emerging, following the spirit of e.g. the Lean Startup framework [11].

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 58–71, 2014.

While newly founded companies may find it easy to start operating in accordance to the Lean Startup ideals, already established companies, especially those that have been formerly a part of the Nokia ecosystem, have been operating with a different mindset. They have been largely focusing on creating software products, which have been delivered in a somewhat traditional fashion, and this is also reflected in their processes and organization, which complicates entering new markets and experimenting with new products.

To create the foundation for the future success of the Finnish software intensive businesses in the new digital economy, largely fueled by the Web and pervasive connectivity in almost all places, a collaborative, industry driven research program has been created, called Need for Speed (N4S) [8]. The project is planned for years 2014-2017, and its budget exceeds 80M€, resulting in an annual budget around 20M€. The program is executed jointly by the industry and academia, and it presently is the biggest national investment in software-related research.

In this paper, we give an insight to the joint goals and concrete actions of the program and discuss the motivations of individual companies that are participating in the program. Moreover, we will also introduce research actions that will be executed during the first year of the program. As an additional contribution of the paper, we address a concrete business goal of the project, so-called Mercury business, where the principles of the Lean startup framework are applied in a more conventional industrial setting, and compare our goals with those of the Lean Startup approach. Moreover, topics such as internal startups and elastic enterprises are also closely related to our approach, and they will be briefly addressed as well.

The rest of this paper is structured as follows. In Section 2, we address agile and lean software development, which reflects the general state-of-the-practice in Finnish – and to a great extent also Global [10] – companies, although there are small deviations [5]. In addition, we provide some background information regarding new software and business approaches that are applicable in the N4S setting. In Section 3, we introduce the goals of N4S, and discuss each research goal separately. Moreover, we also show the big picture of these goals to demonstrate the changes we are aiming at. In Section 4, we provide an insight to the initial goals of the companies that are participating in the program, and cluster them in accordance to the different research themes of the program. In Section 5 we give an extended discussion on our observations so far in the creation of the consortium for the program, as well as point out certain important details. In Section 6, we draw some final conclusions.

2 Background

In the following, we first discuss contemporary software development approaches that are commonly applied in Finnish software companies. Then, we address disruptive technologies that challenge the old ways of working in the field of software. Finally, we briefly introduce the Lean startup approach, which has been an inspiration during the planning of the N4S program.

2.1 2.1 Agile and Lean Software Development

Software and software intensive industry have undergone major advances over the last decades. The transition from slow projects lasting years to the rapid cycles of continuous development and deployment have been dramatic (Fig. 1).

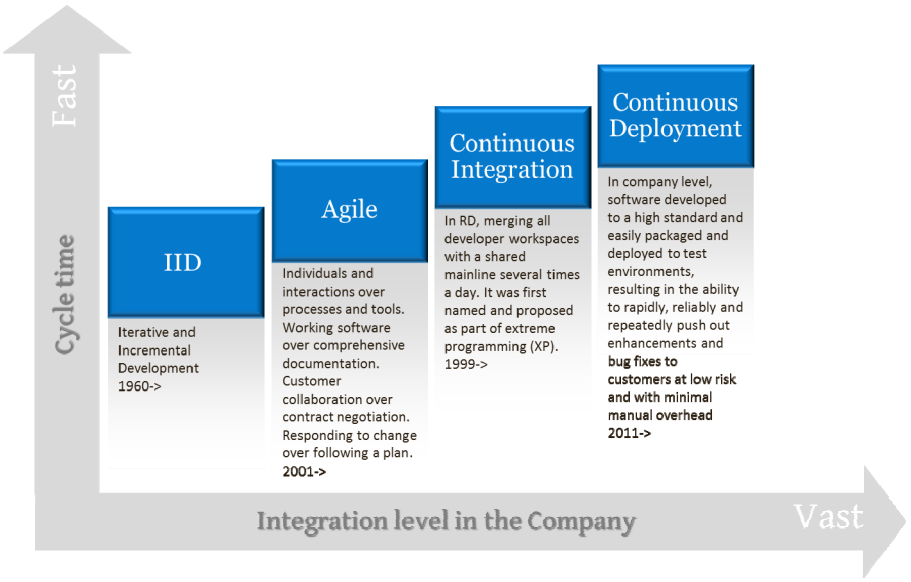

Fig. 1. Agile and lean software development

Iterative and incremental development. Ever since (and probably even before) the introduction of commonly misunderstood Waterfall process [12], iterative and incremental development has been used by software developers to manage risks and uncertainties in software development. By developing software in a piecemeal fashion, where frequent checkpoints can be used to detect anomalies and misinterpretations, the development effort can be more easily managed than by using a big-bang development approach. Consequently, while the rational design process can be used to explain how the development advances [9], in reality it has been customary to conduct at least experiments before advancing too far in the development.

Agile development. In many ways culminating in the Agile Manifesto (http://agilemanifesto.org/), agile software development approaches [3] consist of a wide number of practices where delivering value to a customer is the dominant factor in software development, over following a plan, which had been the prevailing concept early on in many software projects. Various agile methodologies exist, including Extreme Programming [2], Scrum [13], Kanban [1], and Lean software development, which more or less share the underlying mindset but implement the actual actions differently.

Continuous integration. When numerous developers work on the same project, they commonly make changes in the same software components in their own workspaces. When the changes contradict each other, a conflict arises, which need to be resolved by the developers. The key issue of continuous integration is to minimize such conflicts by merging developer workspaces with a shared mainline [6]. Whenever a change to the mainline is made, the whole system is compiled and an automated test run is made to ensure that the mainline remains healthy. It is important to notice that continuous integration is a development related issue, and therefore it mainly concerns R&D of software organizations.

Continuous deployment. While continuous integration is about creating the ability to build a system automatically when even a smallest change has been made, continuous deployment is about creating the ability to deliver the smallest added value to the customers. Obviously, to minimize risks this requires automating all the processes that must be executed to deliver the software to customers, and therefore implementing continuous deployment concerns the whole company.

To summarize, the evolution of the software development approaches has led towards approaches where the step between the development and deployment is being reduced. Hence, an approach referred to *DevOps* emerges, where development is treated similarly to operations, and no distinction between the two is made. To be more precise, DevOps stresses communication, collaboration and integration between software developers and information technology (IT) professionals who are responsible for the operation of the information systems [4]. The promise is that the tighter cooperation results in rapid development and utilization of the software products and services. To reach this target it is common that continuous deployment and/or continuous delivery [7] are used. Moreover, in order to gain benefits from the capability to release rapidly requires that also business goals are defined in a clear and achievable fashion.

2.2 The New Operating Environment

The Internet has rapidly become far more pervasive than it was only a few years ago. At present its transformational effects are spreading into several sectors of the economy and society via new innovations, services, and the emergence and quick success of new companies.

The complexity and competition around the new Internet infrastructure, services and business environment will increase dramatically which will fundamentally change the way software will be developed, deployed and used to reach business goals. The Internet partly already is and will increasingly be the first truly global platform for the digital economy. It will enable significant new business, economic and social opportunities. Consequently, we are facing a fundamental systemic transformations towards a world where digital resources are constantly available on-line, and available for all to use.

These systemic transformations will take many forms. Increasingly, products and services are not developed by a single company but rather by a network of collaborating companies. New, still partially emerging ecosystems and new

competitors will alter industry structures, the public sector, supply chains and many other aspects of today's businesses. Similarly, computing and networking infrastructures, approaches, and processes have changed dramatically over the last years – faster and faster networks, cloud and web technologies providing vast computing capabilities, open source software available free of charge online, Internet of Things (IoT), and open data approaches – and they are all reshaping the digital economy in unforeseen ways and scale.

These new opportunities are increasing ability to gather feedback regarding the use of products, customer satisfaction, and various other aspects that we have commonly overlooked. Such methods are already commonly used in today's software systems to e.g. report bugs – something that is fundamentally associated with software development. However, in the future, there will be similar facilities for other use to help understanding how customers are using products and to create models regarding why. The central concept in the new internet economy is the idea of a minimum viable product or service, which aims at defining the smallest possible implementation that brings added value to customers. Upon delivering the product or service, the focus shifts to creating incremental improvements, so that development cycles can be shortened, progress can be evaluated, and customer feedback and insight can be used to measure the value of the improvement and fed back to development in real-time. Today, game and web service companies are already leading the way towards deep customer understanding to improve gaming experience or to help in e.g. selecting suitable advertisements to show, but we expect that many other fields of computing will be quick to follow. When combined with the ability to rapidly scale operations, the concept resembles that of *elastic enterprises* [14], which we will address later on in the paper.

2.3 The Lean Startup

In the Lean Startup framework [11], so-called Minimum Viable Product (MVP) is a key concept. With MVP the developing organization can find the critical and most valuable features with the customers by experimenting with new iterations in the market.

The core of the Lean Startup approach is to execute a build-measure-learn cycle iteratively. These activities are linked to artifacts, ideas, product, and data. In each iteration, ideas are transformed to products by building them, then, as a product is used, usage patterns are measured, and finally measurement data is used to learn new ideas. The goal of these iterations is to learn what features customers are ready to pay for, and which are not interesting for them.

Customer development is an essential activity in the Lean Startup approach, and it defines how the Lean Startup approach is applied as the company starts to grow. There are four phases that follow each other, 1) customer discovery, 2) customer validation, 3) customer creation, and 4) company building. The goal of customer discovery is to test both problem and product hypothesis, which the customers would like to be solved. Once the hypotheses have been approved, in customer validation phase, the goal is to create a sales roadmap, with a sales cycle that is feasible. It is

possible to iterate between these two phases; agile tactics, such as short releases, simple designs and refactoring commonly play a role in these steps. When the product is good enough, the remaining two phases are executed in more traditional fashion, where business plans based on the product are created and usual market creation activities are executed.

While the lean startup approach defines no particular process or tools that are to be used in software development, in general agile development approaches are assumed to minimize the time from a concept to a prototype that can be experimented with. As for the analysis, basic statistical methods and measures related to business goals are used to study whether the desired results are achieved with the existing implementation. Moreover, so called A/B testing, where different versions are tested in parallel, and the version that is best received by customers will be selected for future use and development, is often applied in this context.

3 Towards Mercury Business

The N4S program has been built around three main themes. These are 1) paradigm change from product business to delivering value at real-time; 2) deep customer insight to improve the hit-rate of businesses; and 3) Mercury business which explicitly aims at finding the new money instead of focusing only on the traditional customers. All these goals build on established practices presented in Fig. 1 earlier, but this time the focus has been shifted from software development view to the business impact created with software.

In the following, these three goals, which are also illustrated in Fig. 2, will be addressed separately in different subsections. However it is important to notice that all of them jointly enable the new breed of software business we refer to Mercury business as described in [8].

3.1 Real-Time Value Delivery

The key aspect of the N4S program is to catalyze a paradigm change from the traditional product-based software business to service-based business where value can be delivered at near real time. Achieving this goal requires careful reconsideration of the mode of operation as well as seamless integration of businesses and research and development – the former provides motivation for the latter, whereas the latter enables new forms of business. Obviously, also technical infrastructure and required capabilities must be established to support the transformation. In addition, special attention is required to maintain the present level of quality, or, better yet, improve the quality experienced by customers by focusing on fewer features but delivering them more rapidly.

To reach the above goals, an architecture that supports the incremental development of systems is needed, where features can be added and removed easily. Moreover, this architecture must be complemented by a continuous integration system that can build and test new versions, implying that automatic and incremental generation of test cases and interpretation of test results are a necessity. Finally, deployment of the software must also be automated, with mechanisms to minimize or eliminate downtime and inconveniences for the users.

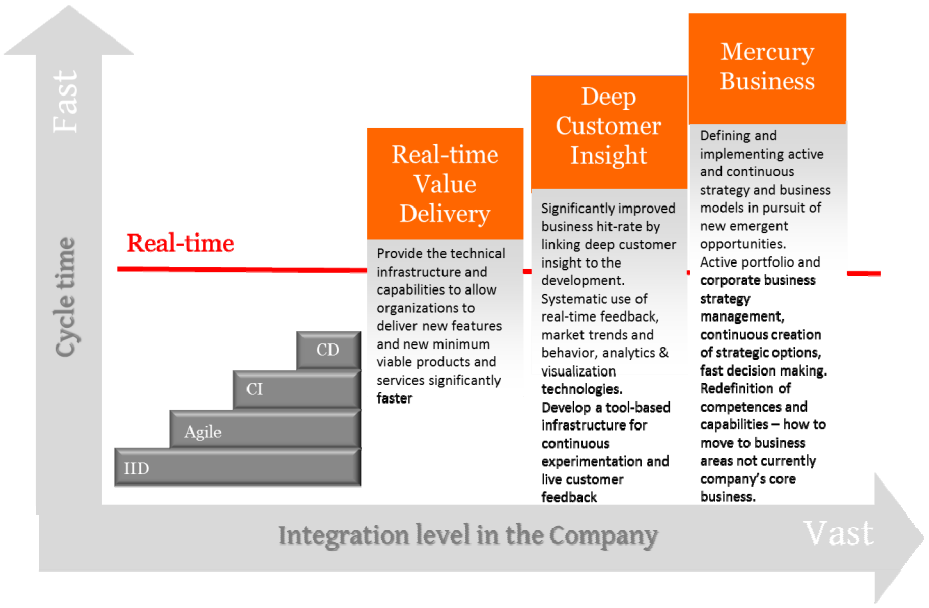

Fig. 2. Real-time value delivery, deep customer insight, and mercury business

3.2 Deep Customer Insight

The goal of deep customer insight is to invent value-creating solutions, and act as a source of inspiration for new products, features, or services that create customer value, which typically stems from the customer contexts and not from the engineering domain. The goal is to quickly gain and assess information regarding the true customer value of potential services, product features, and other possible aspects of user interaction with a service or a product. As a prerequisite, understanding of customer contexts and development opportunities as well as an insight on the ways how customers live and work are needed. The deep understanding of the customers, usage of products and rapid feedback are gathered continuously from the live use of the products, and any possible weak signals.

Conducting live experiments enable studying how the users actually interact with a service or a product. However, successful collection of usage data requires understanding regarding what data to collect. Data that is readily available and simple to collect does not necessarily lend itself to meaningful interpretation in terms of what can be related to the user value of the features or true needs of the user. Therefore, before running the experiments, these experiments should have a defined scope and purpose. One can start the experimentation from simple features and interactions, but the ultimate goal of the program is also to enable experimenting and testing ideas and concepts early in the development – not only after the product or service or hardware for the product exists.

To achieve the above goals, there is a thriving demand for automatic and efficient feedback systems, analytics and visualization. The potential of efficient feedback

systems and analytics of different flavors is huge, as companies realize the importance of understanding their customers' cultural differences and behavior in different situations.

3.3 Mercury Business

By Mercury business, we refer to companies and societies being able to behave like "mercury" finding new grooves where to flow to grow new business. The goal is to enable companies to actively seek new ways to execute their existing businesses, and, perhaps even more importantly, also experiment the options to transform themselves to completely new business areas. The two above goals, real-time value delivery and deep customer insight, are important prerequisites for Mercury business, but there are also other factors that must be considered. For instance, it is obvious that company culture, structure, and leadership must be altered – from individuals and going all the way to organizational structures – to empower everyone to seek new opportunities. Indeed, one important factor is extreme organizational flexibility, where all kinds of changes are made culturally as easy as possible. The ways of working may also change dynamically regardless of the existing organizational structures. These changes are possible e.g. in the Finnish individualistic culture, where extremely dynamical changes in the ways of working are possible.

Finally, while the Mercury business model may change existing products and portfolios, we believe that its ability to totally convert the company into a new business domain is more important. This is what we believe will be an important characteristic for the next-generation software business even in the global scale.

Lean Startup vs. Mercury Business. As already mentioned, Mercury business is closely related to the Lean startup framework, and in many ways the Lean startup has been an inspiration for Mercury business. However, while Lean startup is about the creation of a new company and the definition of its products, Mercury business aims at transforming and extending already existing businesses, which requires a different approach. The main differences between the Lean startup and Mercury business are listed in Table 1.

4 Thematic Analysis of N4S Cases

From program management perspective the work in the N4S program has been divided to 1-4 cases per participating firm. Each of the cases has a case owner from the firm, a research coordinator from a research institution and one or several firms and research institutions working on the tasks related to the case. The cases are expected to impact the participating firm performance in line with the targets of the program.

Table 1. Lean startup vs. Mercury business

Lean startup	Mercury business
No rigid organization; emerging company that is seeking for a form.	Already existing organization that seeks new markets and opportunities; internal startups can be used to separate new effort from already existing business.
Experiment potential products that could be scalable to different markets.	Experiment scaling of existing products (or product derivatives) to new markets, experiment scaling of features in existing products.
Rapid pivoting where old products can be abandoned for better ones.	Whole experiment is about experimenting new opportunities; existing products and markets not risked.
Usually only one product at a time is being considered.	Numerous parallel experiments are possible.
No existing infrastructure for supporting experimenting; built as a part of the product and the experiment.	Established infrastructure for experimenting must be in place.
Build-measure-learn.	Measure-learn-build.

For the purposes of analyzing the 49 cases defined in the beginning of the program for the 26 firms, the program preparation team analyzed the case descriptions provided by the participating firms, extracted key concepts from them and annotated them with on an average three labels. For focused cases one label described well the connection of the case to the targets of the program while some broad cases needed up to six theme labels. Total 23 labels were used the most common ones being as follows. In Mercury Business area: Mercury business model trials (10), Partnering approaches (6), Skills and capabilities (6), and New market opportunity / domain detection (6). In Deep Customer Insight area: Fast feedback / voice of customer (15), Telemetrics – data analysis (10), Customer and business landscape analysis (8), Multidimensional segmentation (6), and Experimentation culture (6). In Real-Time Value Delivery / Continuous Deployment area: Real-time value delivery tooling (21 cases), real-time value delivery (21), and Variability and reuse management (10).

The target of thematic analysis is to identify, which business cases can be clustered together based on shared themes. For this purpose the annotations the 49 business cases were compared by two means. First, the business cases sharing two or more themes were connected together and the resulting graph created with GVisualize is presented in Figure 3. Secondly, the thematic labeling of each case was treated as a vector in a 23 dimensional space. Figure 4 represents connections between cases, whose vector multiplication exceed a threshold value of 0,30.

From the first graph we can identify four clusters of cases:

- Real-time value delivery and real-time value delivery tooling. This is the largest cluster and includes the cases, where the main emphasis is enabling and automating the continuous deployment and value delivery. This represents the first step in building the capabilities of the firm in the program themes.

- Fast customer feedback and data analysis. The cases in this cluster are making use of the data collected from the customer for the second step of the program, i.e. for gaining deep customer insight.
- Customer and business landscape analysis. These cases operate in between the deep customer analysis and mercury business targets of the program. They connect the landscape analysis either with the fast feedback or with mercury business model trials.
- Mercury business models and variability / reuse management. These cases combine creation of mercury business models to the context of product variability and target in creating new business from this combination.

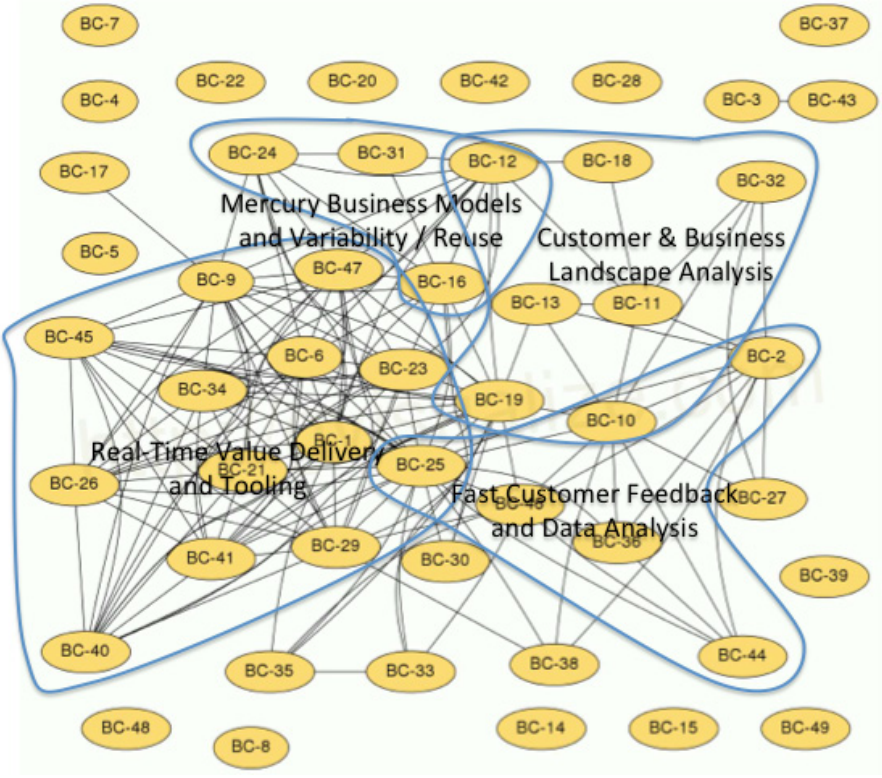

Fig. 3. Cases for year 2014 clustered based on sharing two or more common themes

In addition to the four clusters in Figure 3, there are some cases connected with either the real-time value delivery or tooling and customer feedback (see cases 30, 33, 35 and 38) while they do not form a thematically uniform cluster to the extent the four clusters presented here.

Figure 4 shows three major clusters, which are formed around three key labels, Real-time value delivery tooling, Fast feedback / Voice of Customer and Mercury business trials. These themes were among the most used thematic labels and also parts of the connected themes in Fig 3.

Due to use of the vector multiplication and a high threshold value (0,30) the cases having only one theme label tend to form the core of the clusters. Instead, cases with several thematic labels tend not to match with several labels of another case. And vice versa, cases 15, 22 and 39 in the middle of the clusters in Fig 4 are not connected in Fig 3 as they have only one thematic label.

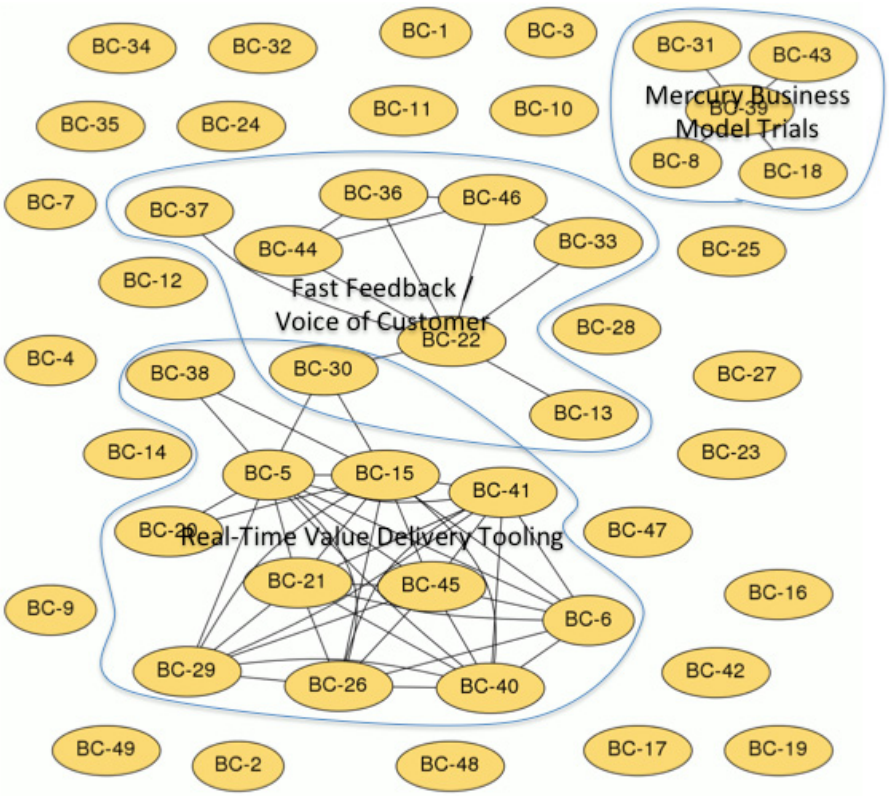

Fig. 4. Cases for year 2014 clustered based on vector multiplication value exceeding 0,30

From the perspective of this paper the empirical part brought up the key themes within the three main themes. Quantitatively we also noticed that majority of the cases start with the first main theme with real-time value delivery, followed by cases focusing on using fast customer feedback and data analysis as the means to gain deep customer insight. In area of mercury business the mercury business model trials were

connected with customer/business landscape analysis or variability/reuse management. In addition, the main theme on Mercury business included several cases focusing on partnering approaches, skills and capabilities, and new market opportunity / domain detection.

5 Discussion

The increasing interest in seeking new markets to face increasing competition requires lean approach to numerous operations. Contemporary software development ideologies – such as Scrum, Kanban, Lean production, and DevOps mentioned above, are building on the possibility to perform small changes that are delivered to the customers as soon as they are completed. While this delivery does not need to take place immediately as the new features are completed, the option to do so is of pivotal importance in Mercury business, as the decision regarding the deployment can be made based on markets rather than technical competences and capabilities – in other words the technical capabilities are extended to business operations. Similarly to software development, the ability to execute new business does not necessarily mean that actions should be taken immediately, but for obvious reasons, such as advertising campaigns, the exact time to go live may be a subject to a strategic, company level decision.

The execution of Mercury business builds on some of the characteristics of the elastic enterprise [14], where five key dynamic properties have been identified that help in scaling businesses in aggressive fashion. These are business platforms, business ecosystems, universal connectors, cloud infrastructure, and sapient leadership. In Mercury business, each business attempt still builds on these properties, but the attempts should be framed as a live experiment. In particular, there must explicit goals that determine whether or not it makes sense to continue the attempt to create new business through scaling the existing business and technical infrastructure.

To summarize, the most important differences between Mercury business and Lean startup arise from the fact that in an already established company, there commonly are assets that the company seeks to benefit from also in the future. Identifying the way and the domain in which the assets become valuable are the key issue of Mercury business. By contrast, in the Lean startup approach, the key question is what assets to build. Another difference is that while a company with existing business can extend its resources to various parallel experiments, in the creation of a startup the focus is commonly placed on the most important aspect. Our claim is that the cost of these experiments will be significantly reduced with the help of highly automated infrastructure providing capability of real-time value delivery with deep customer insight. What is common in both approaches are the elements regarding scalability of assets, be it those that a new company will build or those that already exist. We believe that scaling is the fundamental key characteristic of all successful Internet era businesses.

6 Conclusions

The newly emerged Internet based business operating environment is paving the way towards a new world of business. These days, everyday artifacts and services such as documents, photos, music, videos and newspapers are widely available on the Web. Online banking and stock trading have become commonplace. Various documents that used to be difficult to access, such as municipal zoning documents, government budget documents or tax records, are now readily available on the Web.

To deal with this change, many companies are at the brink of a major shift on how they define their next-generation competitive strategy, new leadership approach and operating processes that would form a strong basis for changing economic conditions. The key question is how the companies could adapt to radically new business conditions and opportunities in real-time or even proactively.

The quantum leap in software development speed by incrementally building and deploying software with real-time customer feedback will facilitate the speed and flexibility needed in the Internet-time business competencies. Perhaps paradoxically, software development, which has sometimes been criticized for slowing down the business, has become a source for rapid innovations. Harnessing this ability to serve strategic business intents requires drastically new approaches. The transformation and radical rethinking which takes companies into totally new markets and enables them to benefit from the most viable business opportunities, are built on the concepts such as new strategic thinking and leadership, rapid development cycles, validated learning, scientific, but cheap live experimentation, and iterative releases with minimum viable products and services.

References

1. Anderson, D.: Kanban – Successful Evolutionary Change for your Technology Business. Blue Hole Press (April 2010)
2. Beck, K.: Extreme Programming Explained, 2nd edn. Addison-Wesley Professional (1999)
3. Cockburn, A.: Agile Software Development, 1st edn., 256 pages. Addison-Wesley Professional (December 2001)
4. Debois, P.: Devops: A software revolution in the making. Cutter IT Journal 24(8) (2011)
5. Eloranta, V.-P., Mikkonen, T., Koskimies, K., Vuorinen, J.: Scrum Anti-Patterns: An Empirical Study. In: Proceedings of 20th Asia-Pacific Software Engineering Conference (APSEC 2013), Bangkok, Thailand, pp. 503–510 (2013) ISBN 978-0-4799-2144-7
6. Fowler, M.: Continuous Integration (2006), `http://martinfowler.com/articles/continuousIntegration.html`
7. Humble, J., Farley, D.: Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education (July 27, 2010)
8. Huomo, T., Järvinen, J., Kettunen, P., Kuvaja, P., Koivisto, A., Lassenius, C., Lehtovuori, P., Lilja, S., Miettinen, S., Mikkonen, T., Münch, J., Männistö, T., Oivo, M., Partanen, J., Porres, I., Still, J., Tyrväinen, P.: Strategic Research Agenda for Need for Speed. ICT SHOK DIGILE (April 22, 2013), `http://www.digile.fi/Services/researchprograms/futureprograms` (last referenced January 2014)

9. Parnas, D.L., Clements, P.C.: A rational design process: How and why to fake it. IEEE Trans. Softw. Eng. 12(2), 251–257 (1986)
10. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J.: The impact of agile practices on communication in software development. Journal of Empirical Software Engineering 13(3), 303–337 (2008)
11. Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Publishing Group (2011)
12. Royce, W.: Managing the Development of Large Software Systems. In: Proceedings of IEEE WESCON, vol. 26, pp. 1–9 (August 1970)
13. Schwaber, K.: Scrum Development Process, In Business object design and implementation. In: OOPSLA 1995 Workshop Proceedings, p. 118. The University of Michigan (1995)
14. Vitalari, N., Shaughnessy, H.: The Elastic Enterprise: The New Manifesto for Business Revolution. Telemachus Press (2012)

The Role of Business Model and Its Elements in Computer Game Start-ups

Erno Vanhala and Jussi Kasurinen

Software Engineering and Information Management
Lappeenranta University of Technology.
P.O. Box 20 FI-53851 Lappeenranta
{erno.vanhala,jussi.kasurinen}@lut.fi

Abstract. In this multiple case study we interviewed six Finnish computer game start-ups to find out what elements are included in their business models. We identified the key elements and used the analytical hierarchy process to rank the elements. We found out that computer game start-ups see their business model as a synonym to a revenue model and/or a business plan. In an in-depth analysis we identified nine key elements (human capital, marketing, key partners, financing, customer relationship, key activities, innovation process, key resources and customer segment) that have operative importance for these companies. These elements are the building blocks of a business model in the computer game start-up domain. The findings provide improved knowledge on how the business models of game start-ups could be constructed.

Keywords: business model, computer games, start-ups, multiple case study, analytical hierarchy process.

1 Introduction

Business models are useful in modern business environments as they allow organizations to understand where their value comes from and how the company in general operates. However, in our earlier study [1] we found out that very little research has been conducted on the role of business models in software companies that could explain their special features and compare their business models to those of other fields, such as mechanical or food industry. Some studies have defined the concept of a business model [2], [3] and some have made observations on software business [4], [5], but there seems to be a lack of research that observes the business model from the software company's point of view instead of categorizing software companies based on their business models. Recognizing this we dived into the business of six computer game start-ups and studied their business models.

These companies build technological solutions, products, not to solve problems, but to give value to customers in other ways, mostly by providing entertainment and experiences. Revenue is not generated directly by the technological solution nor by the experiences offered, but by the business model generating revenue from

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 72–87, 2014.

technology and experiences [6]. As the business varies, it is also probable that the business model must contain variation in parts, relationships and their weighting.

The overall definition of a business model can be described for example by how it captures the way a company functions and creates value and delivers value to the customer and how it converts the customers' responses into profit [7]–[10]. We have already noted [1] that the definition is ambiguous, and different researchers still see the concept of the business model in a different way.

In this study we aim to answer three questions, which have been touched by the literature but not yet adequately answered [1]. The first question *"How do computer game start-ups define the business model?"* digs into the issue of the concept of business model being young, and thus, as the definition of the term is still somewhat unclear [1], [11], the companies may understand it in various ways. With the second question *"What are the elements of the business models of computer game start-ups?"* we aim to identify the pertinent parts that the managers consider as the elements of their business model. The final question is *"How are the elements of computer game business models prioritized?"* On the basis of interviews, we prioritize the elements.

2 Related Research

There has been a lot of discussion of what a business model is, what parts are included and what are not. A common definition is still to be found [11]. Researchers have positioned the concept of business model between business strategy and business processes [2], and it is argued that the business model fills the gap between the two. On one hand, business strategy is a more abstract way to position an organization in the business, and on the other hand, business processes work within the operational level with more detailed ways of doing business. This segmentation is also supported for example in [3], [12], [13]. A business model is more concrete than just the decision to use segmentation, differentiation or cost leadership as parts of the business strategy proposed by [14], yet it is not as concrete as the concept of a business process, which includes detailed processes like management and operational processes. The business model is not a process, but merely description of the steps and key items [11], [15].

Several studies which define business models identify elements that are characteristics to this concept [3], [4], [11], [16]. The variety of elements is great, but the most commonly used ones include for example value production, customers and the revenue model. The variety of included elements has changed during the years, and for example in 2000 it was mentioned in [17] that a business model and a revenue model are complementary but distinct concepts. In more recent studies, the definition has lived on and the revenue model has been included as one element of the business model concept [11]. As the business model concept is closely related to the concepts of revenue logic and revenue model, Sainio and Marjakoski [13] argue that the revenue logic is a part of the business model, and the business model describes who pays and what he gets in return. They position the revenue logic at the strategic level

and use the concept of the business model when describing the steering done at the operational level. Some studies use the term component [3], [11], [18] while some talk about elements [4], [16]. They all still talk about the same thing: parts that form the business model.

The business model concept has been studied in several business areas - like health-care [19], airline business [3] and software business [4]. Software business differs from the other business domains in many ways, as it builds intangible products and services that a user cannot experience directly but through user interfaces [20]. In our literature study [1] we concluded that there were several articles available describing particular areas of the software business, for example, revenue and pricing issues, how the software-as-a-service paradigm is changing the business, what open source and mixed source mean to the business model and what are the difficulties when a software company is expanding to overseas. However, it seemed that no studies existed describing how software companies understand the business model concept, its elements and its use in daily operations.

3 Research Process

In this study we follow the multiple case study research method [21], [22] and the framework developed in [21]. The case study has six steps: defining the strategy, reviewing the literature, developing the case study protocol, conducting a pilot case study, conducting a multiple case study, and developing a conceptual model. Our research strategy is determined by the 3 research questions presented above. Reviewing the literature was already done in our previous study [1]. The development of the case study protocol included the decision to use interviews as the data gathering method and the design of an interview guide. We conducted a pilot case study and determined that the protocol was sound. The analysis produced a conceptual model, which is presented in Section 4. To guarantee the validity of the results, we followed principles derived from [21]–[23]. This included for example choosing the data collection procedures (we used interviews), data analysis methods (we used coding) and avoiding being biased (we had more than one researcher present at most of the interviews and conducting the analysis of the collected data).

In the analysis we used the analytic hierarchy process method (AHP), which is widely used in decision making [24]. AHP has been used in various areas, such as selection, evaluation, benefit-cost, priority, development, resource allocation, decision making, forecasting, medicine, and quality function deployment. Alidi [25] used AHP to measure the initial viability of potential industrial projects. Babic and Plaxibat [26] used AHP to rank companies according to their business efficiency, and Sarker et al. [27] used AHP to find out the relative importance of various types of agility in information system development. The characteristics of AHP include suitability to problems with multiple criteria and attributes [28]. Hafeez et al. [29] determined the key capabilities of companies using AHP with both quantitative and qualitative data. In this study we use AHP in a similar way – as a tool to prioritize results based on qualitative data.

3.1 Data Gathering and Analyzing

We collected and analyzed data from six Finnish computer game start-ups. A majority of them developed mobile games, but there were also experiences in developing PC/Mac, browser and serious games. The study uses data from three interview rounds. The interview rounds one and two provided us with 931 minutes of interview data for background material, and the third round with 507 minutes of data especially aimed for this study. The first round of interviews included team leaders or project managers, the second round upper management or the owner, and the third one interviews with upper management. In most of the interviews, only one company representative was present, but in two occasions there were more than one person from a company. In total nine persons were interviewed. Information of the companies is presented in Table 1.

The actual interview questions were peer-reviewed within the research group before the interviews were conducted. The questions were open-ended, which enabled also free-form discussions during the interviews. The interviews were sound-recorded and transcribed. The focus of the interviews in the first round was to understand the operational level of software development. The second round focused on marketing, innovating and financing, and the third round focused completely on business issues like customers, revenue models, value propositions, and cost structures.

In this study we have built the interview questions over the ideas of the business model canvas (BMC) developed by Osterwalder et al. [30]. This means that the nine elements (key partner, key activities, key resources, value propositions, customer

Table 1. Describtion of the organizations.

	Case A	**Case B**	**Case C**	**Case D**	**Case E**	**Case F**
Size of the organization	4 persons	4 persons	8 persons	3 full time, 1 part time	4 persons	3 persons
Relatedness to games	Makes games	Makes games	Makes games	Makes games	Makes serious games	Makes games
Number of released games	1st one being developed at the moment	First two being developed at the moment	2	1	2 projects being developed at the moment	1st one being developed at the moment
Years in business	Less than 1	Less than 1	Less than 3	Less than 2	Less than 2	Less than 1
Platform / Customer segment	Smartphones	Smartphones, tablets, browser games	Smartphones, tablets, desktop computers	Browser games, smartphones	Browser games, smartphones	Smartphones

relationships, customer segments, channels, revenue streams, and cost structure) of BMC were used as the "seed categories" for the interview questions. These categories were modified during the question set-up to be more suitable for the software business, and also new categories appeared. For example, the weight of the channel category of BMC was decreased and the roles of customers and partners increased, as we saw them more important for computer game start-ups. Our final interview themes included six topic groups for the questions: customer; key partners and resources; business model and value proposition; cost structure, modeling and marketing; organization and industry; and reasons why the company was started. These six main topic groups were covered in the questionnaire with 3 to 7 question items in each group. The final questionnaire form is available online at http://www2.it.lut.fi/projects/SOCES/library.

4 Elements of the Business Model

The topic groups were based loosely on the business model canvas [31]. However, the results indicate that the case organizations emphasize different topics from the ones highlighted in the business model canvas. Some elements match, but some are less important than described in [30].

It was asked from the organization how they have modeled their *business,* to get a rough idea on what they thought about the topic. Case E (interviewed as 1st in the 3rd round) answered that *"Always when things change and such. To be an entrepreneur it is always like going from one crisis to another, but we analyze and go through it."* When asking what tools they used for modeling we got the answers *spreadsheet* and *3rd party analyzers*. After other interviews we understood that the spreadsheet was used to calculate different *revenue model* possibilities, as Case F put it: *"If we put the price like this, and selling is like that, we see how much operating loss we get"*. 3rd party analyzers meant that some public funding partner had required a *business plan* to be supplied with the application letter. So, for these organizations the term *business model* was used to mean a *revenue model* and/or a *business plan*. As the concept of business model in software business is yet to be defined unambiguously [1], we saw that these kinds of interpretations are likely to pop up. This meant that we needed to analyze carefully whether the interviewed case organization talked about the same issues with the same terms than we did. In this study we research business models, not just revenue models or business plans. Although the organizations saw the business model as a narrower issue, we understood their sentiments on a broader scale than just a revenue model.

Another issue to note is the term *customer*. Traditionally companies have been doing business with customers who give them income. With the free-to-play revenue model, games have players who do not give any (direct) revenue to the company. In the free-to-play model the game is distributed free of charge to anyone with a compatible game system. The revenue is gathered through, for example, traditional

online advertising, cross-game advertising, and especially in-app-purchasing, which means that the players can for example use the normal weapons provided with the game or spend money to purchase better weapons or unlock advanced features. This creates the dilemma of who is the customer: all players or only those players who give income? When discussing this with the game companies they saw all the players as their customers – whether they pay or not. Case E saw health-care organizations as well as end-users as their customers. If they put their application to app stores, customers are also gained from there. Because of this, we define the term customer to include all the gamers, not just the ones who pay.

Let us consider two elements of the business model canvas [31]: value proposition and channels. In the computer game context all game companies described the value they offer to players as an *entertaining experience*. The overall goal of many conventional utility-producing software systems is to save time or enhance the efficiency of the user, whereas the game business has the opposite goal. The manager of Case D summarized this phenomenon: *"[traditional software] tries to minimize the time a user needs to spend. With games we try to maximize the time spent, and still keep it entertaining."* This is one of the areas that separate the game business from the conventional software business. The whole value proposition is turned upside-down, and to find similar value propositions, the music, movie and television industry are closer to the game industry than the conventional software business.

In this study we do not concentrate on the value proposition as it was so obvious for the companies – with the slight exception of the serious game maker Case E, which builds entertainment experience but also aims at health-care savings through rehabilitative games. This study concentrates on the business model elements that enable the entertaining experience, as described below with each individual element.

Another different element is the channel used to deliver the product to the customer. The brick and mortar business needs a physical channel to push products to customers, whereas the software industry is moving towards a completely digital distribution of software. For example, mobile games and other apps are purchased and installed via platform-specific digital stores such as Apple's App Store (smartphones) or Valve's Steam (PC workstations). This reduces the time game developers need to use for planning and designing the delivery channel for their products.

4.1 Description of Individual Elements

We used the ATLAS.ti software to code the interviews and the identified nine business model elements that rose from the data. These elements are the parts that enable business for the case organizations and thus impact the producing of the entertaining experience of the game for the customer. Descriptions of the identified elements are presented in Table 2.

Table 2. Descriptions of the indentified elements

Element	Description
Customer relationship	The customer relationship element includes all the communication and data collection that takes place with the customer. There are two ways to collect feedback. Firstly communication, where the company discusses with its customers in Facebook, blogs, forums or any other media that allow communication. Secondly, companies collect indirect feedback through their games; what parts of the game are used most, what are not used. All the efforts aim to improve the product and the experience for the customer. The customer relationship element is also used to improve revenue generation methods.
Customer segment	The customer segment denotes how the organization invests to find the best possible way to reach the customers and what kind of persons there are in the target group. In the area of computer games, and especially in mobile games, this means mostly selecting the platform that provides the highest profit for the money spent on development. It also includes research on customer behavior and market segments.
Financing	Financing is a key area in business, and it means getting external funding (e.g. venture capital or loan from a bank) and direct revenue from the product to run the business. As the cases were start-ups, they mentioned both external funding and building a revenue model to generate revenue from the games. Some companies also mentioned an aim to build a brand from their game characters to start getting revenue from merchandising.
Human capital	Human capital means the people working directly in the company. People can work full-time or part-time. All the companies pointed out how important their workers were. Many mentioned how the company was especially formed around their key persons.
Innovation process	In a previous article [32] we examined how these companies innovated and were creative; meaning what methods they utilized to produce creative parts, like new game concepts and characters. We learned that they saw innovation as an important element in the game business, but the methods they utilized were mostly ad-hoc brainstorming, and no structured methods were used.
Key activities	Key activities mean operations that are required to produce a product. A game company has several key activities. In addition to developing and programming, also graphical designing, 3D modeling and usability testing were mentioned. In some cases also music and sounds were key activities when they were done in-house, but some outsourced it as they did not have resources to do them by themselves.
Key partners	Key partners include the parties that help the organization to, for example, produce and publish the product. This means, for example, outsourced arts, music and sounds. Some cases also listed the publisher as their key partner, but not all as some had the aim to publish games by themselves.
Key resources	Key resources mean the assets the organization sees important and could not manage without. The most important resource was the human capital, but also other things were mentioned. As the organizations mature, they gather intellectual property (e.g. brand, game characters). Even the development tools were seen as key resources, as the companies had invested in them. Hardware was not considered as a key resource.
Marketing	Marketing means all the actions an organization does to get more visibility for their products. The case organizations valued marketing, and in this study marketing includes how companies aim to advertise themselves and their games, what kind of research is done on the topic and with what kind of budget the marketing could be done.

4.2 Ranking of Elements

We used the Analytic Hierarchy Process (AHP) to rank the found elements on the basis of their importance. The AHP consists of several steps. The main idea is to compare alternatives based on a set of criteria to reach out a goal set beforehand [24], [28]. The goal can be for example choosing the best candidate to vote in presidential elections. After the goal has been set, there are probably alternatives already available, as there is usually more than one candidate for the presidency. Then the decision about the criteria, such as age, opinion about climate change and gun laws is made.

After the initial requirements have been set, a comparison is done. In this study the comparisons were done by the authors of this article based on the gathered data. Comparisons mean that every alternative is compared to each other according to every criterion. This means that there will be N*(N-1)/2 comparisons done with every criterion, where N means the number of alternatives. In our case this means 9*(9-1)/2=36 comparisons per criterion. The comparison is done with numbers 1, 3, 5, 7 and 9. 1 means equal importance and 9 absolute importance, 3 (moderate), 5 (strong), 7 (very strong) being between these opposites. It is also possible to use numbers 2, 4, 6 and 8 if the jump between, for example, 3 and 5 is seen too large. Invert values are used to show the importance on the opposite side.

Based on these comparisons NxN – 9x9 in our case – matrixes are produced and their eigenvector is calculated (Tables 3 and 4). On the basis of these eigenvectors and the weights of criteria, the final value can be calculated by multiplying these two. These values are used when the actual decision making (e.g. prioritizing) is done. The weight of a criterion can be calculated through the same process as the eigenvectors for the criteria. We have used equal weight for each criterion.

Table 3. Matrix produced from Case A data.

	IP	F	CR	CS	M	KP	KA	KR	HC
Innovation process (IP)	1	1/3	1/3	3	1/5	1/3	3	1/5	1/7
Financing (F)	3	1	3	3	1/5	1	3	1/3	1/5
Customer relationship (CR)	3	1/3	1	3	1/5	1	3	1/3	1/5
Customer segment (CS)	1/3	1/3	1/3	1	1/7	1/5	1/3	1/7	1/7
Marketing (M)	5	5	5	7	1	3	5	3	1/5
Key partners (KP)	3	1	1	5	1/3	1	3	1/3	1/5
Key activities (KA)	1/3	1/3	1/3	3	1/5	1/3	1	1/5	1/5
Key resources (KR)	5	3	3	7	1/3	3	5	1	1/5
Human capital (HC)	7	5	5	7	5	5	5	5	1

Table 3 is a 9x9 matrix which shows how Case A sees Financing as moderately more important (3) than the Innovation process and strongly less important (1/5) than Marketing.

After a matrix has been formulated, it is then squared several times to get more accurate results. In our case, after four multiplications we got three static decimals to eigenvectors, which are presented in Table 4.

Table 4. Eigenvector calculated from the matrix presented in Table 3.

Innovation process	0.038
Financing	0.075
Customer relationship	0.058
Customer segment	0.020
Marketing	0.205
Key partners	0.070
Key activities	0.031
Key resources	0.137
Human capital	0.365

These values are now the weights of different elements for Case A. The same calculation was done to every case and the total values were calculated by multiplying the eigenvalue matrix with vector $[1/6\ 1/6\ 1/6\ 1/6\ 1/6\ 1/6]^T$.

AHP does not limit the number of alternatives or the criteria. The criteria can also be divided into sub-criteria if needed. With a consistency ratio and a consistency index it is also possible to check whether the judgment is valid [27], [28]. The process of calculating consistency is described thoroughly in [33].

All the case organizations saw themselves as start-ups, but with some elements they had different weights based on their experiences in the field. The overall ranking and importance is shown in Table 5. Each weight reflects the importance of the specific element, and the weights are relative to each other.

Table 5. The ranking of business model elements based on the analytical hierarchy process. The three most important elements are highlighted with inverted colors and the least important in gray.

Rank	**Element**	**Weights**						
		Case A	Case B	Case C	Case D	Case E	Case F	Total
1	Human capital	0.365	0.318	0.267	0.265	0.350	0.317	0.314
2	Marketing	0.205	0.085	0.035	0.114	0.202	0.209	0.142
3	Financing	0.075	0.203	0.135	0.135	0.056	0.107	0.118

Table 5. *(continued.)*

4	Key partners	0.070	0.157	0.185	0.089	0.112	0.068	0.113
5	Customer relationship	0.058	0.050	0.099	0.235	0.122	0.091	0.109
6	Key resources	0.137	0.039	0.086	0.027	0.024	0.038	0.059
7	Key activities	0.031	0.075	0.095	0.042	0.035	0.062	0.057
8	Innovation process	0.038	0.056	0.055	0.054	0.030	0.086	0.053
9	Customer segment	0.020	0.017	0.042	0.040	0.069	0.022	0.035

Based on the empirical data, the most important element was human capital. The companies argued that *"people are the only thing that matters",* (CEO, Case A) and *"people are the only resource a game company can have",* (CEO, Case C). No other element was seen as important, and this is natural as it is a question of intangible products and start-up companies.

There was some variation between the case organizations as regards marketing. For example, most of the organizations saw marketing as an important element that they had no experience and skill of. *"We have been going with the idea that we are unknown – invisible – and we don't have marketing know-how. The first games are exported to different countries via a publisher, who then gives us the coverage",* (CEO, Case D). However, the oldest organization, Case C, described it as an element that was no longer important. *"In the beginning we had lot of marketing and we had our own marketing manager... But now we have noted that in the end marketing plays quite a small role... maybe even more important [than cross-promotion] is the word-of-mouth.",* (CEO, Case C). Mobile game marketing was seen a bit as a black hole as there was no guaranteed way to get a game to become the editor's choice or to any similar promotion position. This led Case C to scale down the marketing efforts. They also trusted their publisher and had already gained success with games, which is something that the other case organizations were still aiming at.

Financing was another element that the companies saw differently. Case B had the most unique way of funding. Where the other organizations had been using personal savings, getting grants and financial support, Case B had chosen to take a loan from a bank: *"To our joint stock company we are applying for a loan... approximately two times 30k euros... so that we can pay a salary to ourselves from the beginning"*, (CEO, Case B). None of the other organizations mentioned anything about loans, but trusted that they would be able to survive with support money to gain revenue from their games. Free-to-play was the dominating revenue model. Only Case E, which made serious games, mentioned that they were going to license their products to health-care organizations. The rest utilized free-to-play at least to some extent. Some used the best of both models, as Case C described *"Both games started as pay-to-play [later free-to-play] and they also had the in-app-purchasing option straight from the beginning",* (CEO, Case C).

Key partners were also seen important, as for instance only three of the case companies mentioned that they could actually do the whole game with their own resources, and one of the organizations, Case E, mentioned that *"we would outsource if we had the money"*. Most of the companies outsourced at least music and sound. The publisher was also seen as a key partner, but some companies were considering not using a publisher in their future projects. Yet, key partners were not thought as important as the core employees of the companies. The main sentiment in the companies was that they would try to improve their own output, and beyond that, outsource the rest of the work. *"Voice-overs have been purchased from the US",* (CEO, Case C). *"We have an art studio [partner] in Bulgaria... ...from them we get high level graphical assets",* (CEO, Case D).

Also customer relationship divided opinions. For example, Case B, which had not yet released anything, had not thought about getting customer feedback and steering their game development towards the gamers' ideas: *"We do not see it as a problem [understanding customers]... when we get something out, we need to take opinions and getting feedback from blogs and forums",* (CEO, Case B). Case D saw customer relationships as more important and said that they were going to answer the gamers' questions and had already implemented some of the ideas which they had got from the gamers. *"When our users give comments, feedback or questions, we answer every one of them",* (CEO, Case D). Case E, which worked with serious gaming, told that for them customer relationships were important, as they needed to be in close connection with medical staff and be able to discuss with doctors and other health-care people to be able to push their games to health-care use. *"We keep close contact with health-care divisions. We have been discussing and negotiating with all the responsible directors and have had meetings with physiotherapists... [through these discussions] we get those pilot patients"*, (CEO, Case E).

With the exception of Cases D and E, all the other had decided to use third-party tools to build their games. Most commonly this meant full game engines, such as Unity 3D. Their idea was to be able to build games in rapid progression, spending months rather than years in development. *"The first version was a plain C++ OpenGL. After that we tried the C++ and Marmalade combo. It made possible for us to have multiplatform software, it abstracted all the interfaces. It was awkward, too. So, after one year of thinking we have now done with Unity in two months more than all the previous work combined"*, (Developer, Case A). Case D had a slightly different approach as they build browser-based games that communicate with a back-end solution, which was seen as one of their key resources. *"We have now developed it for more than a year, so it [backend solution] is our key resource"*, (CEO, Case D).

All the case organizations mentioned the same kind of key activities, including developing a game, drawing graphics, testing the game, promoting the company, and getting grants. User testing was mentioned in many cases as the most important testing activity. As the games needed to provide good experience, the testing feedback from users was considered very important, and was mentioned several times. *"The first step is to press the play button in Unity... ...but a developer can be blind to his work, so the next step is to compile it to a test device and give it to someone who has no money involved in it"*, (CEO, Case A).

The innovation process is discussed in detail in [32]. Generally innovation and creativity are needed when building a game that gives a customer an experience. The case organizations had their own ways of supporting creativity. They used for example idea pitching and brainstorming where all the members of the company had the possibility to tell about their ideas, and subsequently, if the idea was considered feasible, a prototype could be built.

The customer segment was seen very straightforward for the case organizations, as the application store of the target platform (for example Apple's App Store) was the most important release channel, with the exception of Case D and Case E. Case D used HTML5-based technologies and had built their own back-end solution to support their browser-based games and a broader customer segment. Case E developed health-care related games which limited their customer segment, but they had also thoughts of selling their serious games in app stores. *"In the mobile world the basic app could be offered for free, but not our advanced thing. Not a chance, since it has all the hardware and other things"*, (CEO, Case E). Case E also saw the customer segment as more important than the other companies, as it needed to work with different health-care organizations to find customers.

4.3 Summary of the Findings

In the beginning we set three research questions: *"How do computer game start-ups define the business model?", "What are the elements of the business models of computer game start-ups?"* and *"How are the elements of computer game business models prioritized?"* We found answers to all these questions.

For the first question we found out that the game companies described the business model slightly differently than what they actually applied in their daily operations. They described marketing and financing as the key parts of their business, but in the analysis the human capital emerged as the most important element – yet it was not identified through talking about *business*, but instead through *key resources*. We interpreted that the companies used the term business model when talking about their revenue model. As the academic literature includes for example the technical platform or channel [4] as elements of the business model, it seems that there is a distinction between the academic and practical definition of the term.

The importance of human capital was significant. As this study has focused on start-ups, it is clear that a company is focused heavily on the persons who founded it. Several company leaders said that people were the only thing that really mattered, and for example specific development tools, which may have cost thousands of euros, were not seen as important, although they would ease the development and fasten the release of the game.

Today's computer games, especially for mobile platforms, are more and more delivered through digital stores. We did not find any evidence that the companies had difficulties in delivering their games. App Store and similar digital software markets ease the delivery process significantly compared to the situation where software is delivered with physical packages. The problem was not in delivering the game but in reaching the awareness of gamers.

For the second research question we identified 9 elements. Human capital, key marketing, key partners, financing, customer relationship, key activities, innovation process, key resources and customer segment were seen as elements that enable business leading to the entertaining experience of a computer game.

As an answer for the third research question we prioritized the elements with the analytical hierarchy process and found out that the start-ups considered human capital as the most important element of their business model. Marketing and key partners were also considered important.

5 Discussion

This article concerned the application of business models in game industry startup-companies. In the literature we find numerous articles describing the elements of the business model; for example [3] gives an extensive list of these articles. The elements of the business model were gathered from several different industries, and a few studies [4], [16] which described the business model elements used in the software industry were found. Yet, we did not find all of these elements in our studied organizations. We identified nine elements from game companies, which were similar to the identified elements mentioned in previous studies, but even then they were not a complete match. This supports our view that we cannot describe the business model concept by its elements without taking the business domain into account. Our opinion is that we can discuss business models in two ways: A) by using the more abstract concept positioned between the concepts of business strategy and business processes, as presented in [2], [3], or B) by defining the elements that are used in that specific business model. The latter can be very specific, as even not all software business models include the same elements. According to our view, for example the conceptual framework presented in [2] is too abstract to be utilized by start-ups. In this study we concentrated only on computer game start-ups and thus the findings can be applied in the computer game industry and to some extent in other software business, as the computer game industry has similarities with the traditional software industry. It seems that it is not possible to define the concept of business model comprehensively with the elements discovered in previous studies, or at least different elements have very different weights in different business areas. For example, in this study we found out that the distribution channel is not important for computer game companies. The channel is something that does not have to be concentrated on at the moment when Apple's App Store and Google's Play store dominate the mobile markets. On the other hand, human capital and key partners were seen as important elements, but for example Schief and Buxmann [4] do not mention these in their framework.

Besides the theoretical findings presented in this article, the aim was also to help computer game start-ups. This article provides knowledge on what are seen as important elements in the starting computer game business. This may give new ideas to other start-ups, who might not have noted all the issues presented in this article.

We studied six computer game start-ups in Finland. This means that the sample size was small and homogeneous. However, all the companies were aiming at the

international markets with their products, the companies covered different release platforms and genres, and were developing games as their main source of income, so the companies did have variance and were representative organizations of the games industry. We had four different interviewers to avoid interviewer bias, two people conducting the data analysis to avoid observational bias, and the article was discussed extensively with three people familiar with the data to avoid personal bias. Although the findings were consistent throughout the study, further research is required for a better validation of our findings. In addition, the results of qualitative studies should be considered as suggestions or practice-based recommendations outside their original scope and environment.

6 Conclusion

In this study we observed six computer game organizations and how they had built business around their software products – games. All organizations were start-ups and they were still small in size and had limited experience in the field of software business. We performed a multiple case study to find out what the organizations were doing in practice. We used the analytical hierarchy process to prioritize the key business model elements found in the data.

We discovered nine elements that are crucial when starting a computer game business: human capital, marketing, key partners, financing, customer relationship, key activities, innovation process, key resources and customer segment. We found out that the case start-ups weighted the human capital as the most important element in their business. Their understanding of the concept of a business model was greatly focused on the revenue model and was not in line with the academic version of the concept. The organizations also considered for example the distribution process as straightforward and did not see it as an important part of their business, as described in previous studies. Our assessment on this observation is that this feature is a unique part of the mobile game business, and is different from the traditional brick and mortar industries, even from most areas of the software industry.

This led us to the more theoretical finding that the business model as a concept is not completely defined with elements that are transferable between different areas of industry. For each industry, business models are comparable only in specific cases, like mobile games, where all the organizations utilize similar elements.

Our future research will focus on the validation of the weights of the computer game business model elements with a larger number of organizations and studying the key elements more thoroughly.

Acknowledgement. This study was partially funded by the European Union Regional Development Grant number A32139 “Game Cluster” administered by the Council of Päijät-Häme, Finland, and the organizations funding the related research project.

References

[1] Vanhala, E., Smolander, K.: Business Model - What is It For A Software Company? - Systematic Mapping Study. In: Proceedings of the IADIS International Conference e-Commerce 2013, Prague, Czech Republic (2013)

[2] Al-Debei, M.M., Avison, D.: Developing a unified framework of the business model concept. Eur. J. Inf. Syst. 19(3), 359–376 (2010)

[3] Morris, M., Schindehutte, M., Allen, J.: The entrepreneur's business model: toward a unified perspective. J. Bus. Res. 58(6), 726–735 (2005)

[4] Schief, M., Buxmann, P.: 'Business Models in the Software Industry. presented at the 2012 45th Hawaii International Conference on System Sciences, Hawaii, USA, pp. 3328–3337 (2012)

[5] Valtakoski, A., Rönkkö, M.: Diversity of business models in software industry. In: Tyrväinen, P., Jansen, S., Cusumano, M.A. (eds.) ICSOB 2010. LNBIP, vol. 51, pp. 1–12. Springer, Heidelberg (2010)

[6] Yuan, Y., Zhang, J.J.: Towards an appropriate business model for m-commerce. Int. J. Mob. Commun. 1(1), 35–56 (2003)

[7] Casadesus-Masanell, R., Ricart, J.E.: From Strategy to Business Models and onto Tactics. Long Range Plann. 43(2-3), 195–215 (2010)

[8] Osterwalder, A., Pigneur, Y.: An e-Business Model Ontology for Modeling e-Business. EconWPA, Industrial Organization 0202004 (February 2002)

[9] Teece, D.J.: Business Models, Business Strategy and Innovation. Long Range Plann. 43(2-3), 172–194 (2010)

[10] Wirtz, B.W., Schilke, O., Ullrich, S.: Strategic Development of Business Models: Implications of the Web 2.0 for Creating Value on the Internet. Long Range Plann. 43(2-3), 272–290 (2010)

[11] Zott, C., Amit, R., Massa, L.: The Business Model: Recent Developments and Future Research. J. Manag. 37(4), 1019–1042 (2011)

[12] Magretta, J.: Why business models matter. Harward Bus. Rev. 80(5), 86–92 (2002)

[13] Sainio, L.-M., Marjakoski, E.: The logic of revenue logic? Strategic and operational levels of pricing in the context of software business. Technovation 29(5), 368–378 (2009)

[14] Porter, M.E.: Competitive strategy: techniques for analyzing industries and competitors: with a new introduction, 1st free Press edn. Free Press, New York (1998)

[15] Amit, R., Zott, C.: Value creation in E-business. Strateg. Manag. J. 22(6-7), 493–520 (2001)

[16] Weiner, N., Weisbecker, A.: A Business Model Framework for the Design and Evaluation of Business Models in the Internet of Services. In: Proceedings of the 2011 Annual SRII Global Conference, pp. 21–33 (2011)

[17] Amit, R., Zott, C.: Value Drivers of e-Commerce Business Models (2000)

[18] Hedman, J., Kalling, T.: The business model concept: theoretical underpinnings and empirical illustrations. Eur. J. Inf. Syst. 12(1), 49–59 (2003)

[19] Hwang, J., Christensen, C.M.: Disruptive Innovation In Health Care Delivery: A Framework For Business-Model Innovation. Health Aff (Millwood) 27(5), 1329–1335 (2008)

[20] Chesbrough, H., Spohrer, J.: A research manifesto for services science. Commun. ACM 49(7), 35 (2006)

[21] Gable, G.G.: Integrating case study and survey research methods: an example in information systems. Eur. J. Inf. Syst. 3, 112–126 (1994)

[22] Meyer, C.B.: A Case in Case Study Methodology. Field Methods 13(4), 329–352 (2001)

[23] Klein, H.K., Myers, M.D.: A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS Q. 23(1), 67 (1999)
[24] Vaidya, O.S., Kumar, S.: Analytic hierarchy process: An overview of applications. Eur. J. Oper. Res. 169(1), 1–29 (2006)
[25] Alidi, A.S.: Use of the analytic hierarchy process to measure the initial viability of industrial projects. Int. J. Proj. Manag. 14(4), 205–208 (1996)
[26] Babic, Z., Plazibat, N.: Ranking of enterprises based on multicriterial analysis. Int. J. Prod. Econ. 56-57, 29–35 (1998)
[27] Sarker, S., Munson, C.L., Sarker, S., Chakraborty, S.: Assessing the relative contribution of the facets of agility to distributed systems development success: an Analytic Hierarchy Process approach. Eur. J. Inf. Syst. 18(4), 285–299 (2009)
[28] Chen, M.K., Wang, S.-C.: The critical factors of success for information service industry in developing international market: Using analytic hierarchy process (AHP) approach. Expert Syst. Appl. 37(1), 694–704 (2010)
[29] Hafeez, K., Zhang, Y., Malak, N.: Determining key capabilities of a firm using analytic hierarchy process. Int. J. Prod. Econ. 76(1), 39–51 (2002)
[30] Osterwalder, A., Pigneur, Y., Tucci, C.L.: Clarifying Business Models: Origins, Present, and Future of the Concept. Commun. Assoc. Inf. Syst. 15, 1–25 (2005)
[31] Osterwalder, A.: Business model generation: a handbook for visionaries, game changers, and challengers. Wiley, Hoboken (2010)
[32] Vanhala, E., Kasurinen, J., Smolander, K.: Design and Innovation in Game Development; Observations in 7 Small Organizations. presented at the ICSEA, Venice, Italy (2013)
[33] Alonso, J.A., Lamata, M.T.: Consistency In The Analytic Hierarchy Process: A New Approach. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 14(04), 445–459 (2006)

Following the Money: Revenue Stream Constituents in Case of Within-firm Variation

Matti Saarikallio and Pasi Tyrväinen

matti@saarikallio.net, pasi.tyrvainen@jyu.fi

Abstract. The idea of this paper stems from the perception that the concept of revenue stream requires clarification and further division to be applicable to businesses with high internal variation in their methods of capturing revenue. Current study sets out to investigate the concept of revenue stream through an overview of previous literature and a case study to demonstrate how revenue streams of a b2b (business-to-business) software service firm can be analyzed by elaborating the concept further. The aim is to answer the following research questions: 1) What are the relevant constituents of the revenue stream concept within a b2b software services firm? 2) How revenue stream as part of the business model can be analyzed within a firm? This exploratory study contributes to the business model literature by investigating the concept of revenue stream and revenue stream type as managerial tools to better understand the business under investigation. The study further attempts to contribute to the decomposition of the revenue stream concept by exploring its constituents in the context of b2b software business. It is suggested that revenue streams in this context should be approached based on sub-component level analysis where the reason and source dimensions create a matrix of analysis cells from which revenue stream types emerge based on similarities in the method of the revenue streams. Based on previous literature and empirical study, it is further suggested that the revenue stream has three main constituents or sub-components: 1) the source of revenue, 2) the reason for revenue and 3) the method of revenue.

Keywords: Business model, revenue stream type, software service company, b2b, source of revenue, reason for revenue, method of revenue.

1 Introduction and Background

1.1 Business Model Research

Through experience, business practitioners have mental models about their business, but such mental model can only be communicated and modified once it has been made explicit as a business model [1]. Research about business models has been around for a long time in the domain of software firms. Still, research knowledge about business model is disjointed and unclear [2]. While there is not yet a common understanding, ontologically business model has been suggested to reside in the middle ground between business strategy and business processes [3].

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 88–99, 2014.

There are various ways to conduct research relating to business models. Research sub-domains can be divided into definitions, components, taxonomies, representations, change methodologies, and evaluation models [4]. The goal of component research is to further decompose the business model concept into its fundamental constructs [4].

The business model concept and its sub-components are used often as a tool to plan and define the business model of new startups. For example Mahadevan [5] uses the term revenue stream to mean the plan for revenue generation. However, business model can also be used to analyze an existing established firm to gain understanding about the de facto business model in place. Such an approach has been taken for example by Rajala, Rossi & Tuunainen [6] in their software business evaluation framework. The idea for the current paper stems from the challenges in analyzing an existing firm's business model's revenue streams when the firm under investigation has multiple customers and offerings with high variability in revenue stream configurations.

1.2 Revenue Stream

Most business model conceptualizations include a financial aspect relating to the money that flows into the company. Business model literature is filled with various terms used for these aspect such as: revenue stream, revenue, sources of revenue, revenues, revenue model, revenue mix, revenue side of the business, revenue source, revenue logic, revenue earning logic, revenue mechanism, income model and earnings logic[5][7][8][30][9][10][6][18][11][23][12][13][14][15]. Table 1 summarizes the terms and what they are suggested to mean in the context of business model. The same unclarity that exists for the business model appears to be present for the revenue related sub-components as well. There seems to be a common theme, but not a clear agreement on the terminology.

Zott and Amit [16] have suggested that revenue model complements a business model design in similar way as pricing strategy complements product design. This can be a useful analogy but in the same way as business model is quite an abstract concept when compared to product design, revenue model is very much as abstract compared to pricing strategy. Revenue stream on the other hand seems to have potential to be defined as a more tangible and measurable object of study as it can be reduced to the concrete idea of money flowing into the company. For this reason of seeking conceptual clarity, this paper focuses on the revenue stream as the main concept of business model and also adopts the approach used in the business model canvas concept suggested by Osterwalder and Pigneur [17].

Table 1. There is a multitude of partially overlapping revenue related business model concepts

Author	Business model component	Description
Mahadevan (2000)	Revenue stream	The plan for revenue generation
Weill, Vitale (2001)	Sources of revenue	Description of source of revenue and how realistic they are.
Alt, Zimmermann (2001)	Revenues	The "bottom line" of a business model.
Stähler (2002)	Revenue model	From what sources in what ways is the revenue generated.
Stähler (2002)	Revenu mix	The sum of all the sources of revenue the firm has.
Magretta (2002)	Revenue side of the business	How is money made in this business.
Afuah, Tucci (2003)	Revenue source	Where is the income coming from, who pays when and for what value and also what are the margins and their drivers for each market.
Rajala et al. (2003)	Revenue logic	The way the software business generates its revenue and profit.
Osterwalder (2004)	Revenue model	The way company makes money through a variety of revenue flows.
Gordijn et al. (2005)	Revenue earning logic	Generating profitable and sustainable revenue streams.
Chesbrough (2007)	Revenue mechanism	How will the firm be paid for the offering.
Rédis (2009)	Income model	Sources of income generated by the company.
Nenonen, Storbacka (2010)	Earnings logic	How the firm yields a profit from its operations.
Schief, Buxmann (2012)	Revenue	Group revenue deals with the pricing model and financial flows.
Ojala, Tyrväinen (2012)	Revenue model	How a firm collects revenue through options that a firm may offer to customers.

1.3 Aims of the Paper

Thus, the current study aims to contribute to the research domain of business model components by investigating a sub-component referred to as revenue stream. For example Osterwalder [18] sees the revenue streams as one of the key parts of the business model. He uses the broader term revenue model to mean a collection of revenue streams within a company.

Osterwalder and Pigneur [17] have claimed that a business model can have two different types of revenue streams, namely transaction revenues and recurring revenues. While this is true in simple business models it is highly unlikely that such a simplification is enough to fully explain the revenue stream sub-component of the business model in the more complicated case.

Shafer, Smith, Linder [19] cite a study by Linder and Cantrell [20] which states that 62 % of executives had a difficult time describing how money is made in their company. This could indicate the complexity of the typical revenue models or that there is a lack of proper conceptualization. Either way this supports the relevance of the current paper's interest area.

This paper aims to clarify the revenue stream component by evaluating the revenue model of a case company which has multiple and variable revenue stream configurations and suggest an answer to the question: 1*) what are the relevant constituents of the revenue stream concept within a b2b software services company*. Further the study attempts to answer the question: 2) *how revenue stream as part of the business model can be analyzed within a firm.*

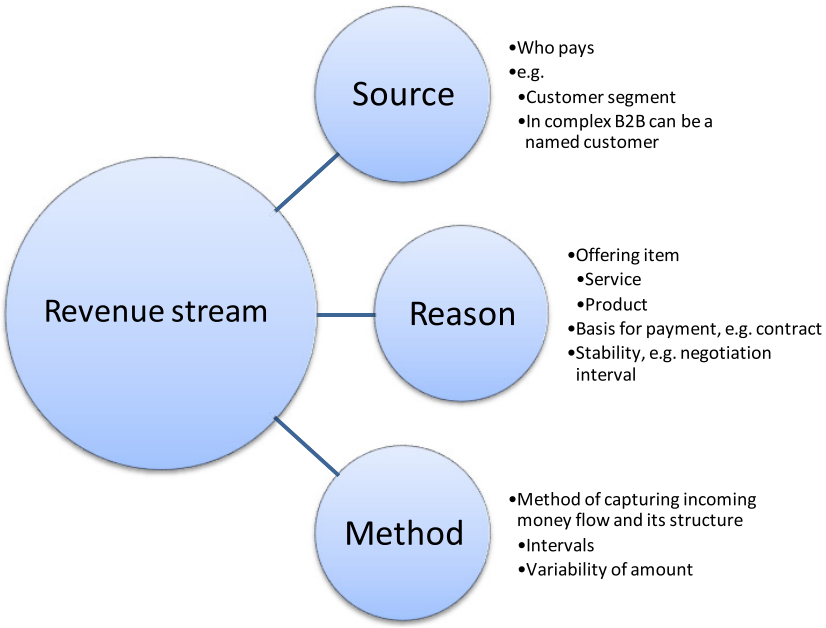

Fig. 1. Suggested decomposition of revenue stream

1.4 Revenue Stream Framework

Framework to analyze the case study data is suggested based on existing literature. It includes three key parts that must be addressed to explain a revenue stream. These constituents are the source of the revenue stream, the reason for the revenue stream and the concrete description of the method of capturing the revenue which is called here the method of revenue. This framework builds upon Rajala, Rossi, Tuunainen and Vihinen [21] who suggest that approaches for capturing revenue can have differences in methods of pricing, sources of revenue and the products and services being sold. Similarly in context of business model innovation, revenue model innovations include as key parts offering reconfiguration and pricing models [22]. Chesbrough [23] uses the term revenue mechanism which is by definition comparable to method of revenue. Figure 1 illustrates the suggested model.

2 Methodology

2.1 Exploratory Case Study

Yin [24] suggests using a case study design when trying to answer how or why questions and attempting to cover contextual conditions relevant to the phenomenon. In the current study attempt is made to understand how revenue stream as part of the business model can be analyzed in the context of a specific b2b software service firm.

When seeking to clarify the concept of revenue stream and related sub-components, it was necessary to analyze the patterns underlying them and it was required to gain an in-depth understanding. Qualitative research approach was chosen to improve understanding of the investigated phenomenon [25]. The chosen research strategy was a single case study in a company that is considered a representative example, because it had enough complexity and variation in forms of multiple revenue stream combinations. Because a case study research strategy focuses on understanding the dynamics present in a single setting [26], it was a good approach in exploring the business model sub-component and how it can be analyzed in a real-life setting in a within-firm context. Thus, research strategy was that of a single case study. Eisenhardt [26] has suggested that instead of selecting cases at random, extreme examples are appropriate when seeking to extend theory, which is the goal in the exploratory research that this paper undertakes. Because a lot of the existing literature considers cases where there is one revenue model per business model, an extreme example deviating from the norm would be a case with multiple co-existing revenue models and high within-firm variation in the revenue streams. The selected case meets these criteria.

2.2 Case Firm

The chosen case firm operates in the telecom operator software market. This market had only 196 companies offering software product or service offerings in 2006 with a volume just under $30 billion [27]. Using the terminology from Luoma, Frank &

Pulkkinen [27] the firm can be classified as a generic telco vendor. It can be predicted that this kind of firm would have a lot of variation in the revenue streams, because of the breadth of operations.

The analyzed case firm serves telecom operator customers by offering BSS (business support system) solutions. The solutions typically contain a service contract which is one side of the business and making continuous customer specific modifications is an additional way to generate revenue. New customers are a rare occasion and typically some sort of penetration pricing is used for initial deliveries. This is possible due to heavy vendor lock-in that is gained once the delivery is completed. The investigated firm has out of 150 people about 80 working in the investigated business unit. It was established in 1995 and has international customers. Relevant customer count is around ten, but three customers produce majority of the revenue. The firm is organized into customer serving teams with minor common functions. R&D, and marketing and sales departments are manned in ad-hoc manner and no organization exists for these functions. This has given rise to a very variable culture across customer serving teams and most interestingly to this paper it has given rise to a multitude of methods for revenue capture. The complexity of the case makes it a useful context to investigate revenue stream variation.

2.3 Data Collection

The main portion of the data was gathered using semi-structured interviews. Twelve people in corporate and business unit management and account management positions were interviewed to find out the current revenue streams of different customer accounts and the various offerings and revenue capture methods for each. The interviews lasted from one to three hours each and some were conducted in two separate sessions, because of scheduling challenges. In addition to revenue model specific questions, the understanding of the case was further widened by questions relating to general business model utilizing the business model canvas framework [17]. All interviews were recorded and transcribed. The interviews were scheduled close to each other during a period of one month. Close scheduling was done in order to avoid participants from influencing each others' answers. Some details were clarified by additional short discussions to avoid false interpretations.

In addition to the interviews, access was gained to written materials, mainly contracts and offers made by the case firm. This helped to solidify the actuality of contractual relations with case firm's customers in situations where the informants were unable to remember the details in full.

Data was analyzed using qualitative content analysis method with three analytical procedures of summary, explication and structuring as suggested by Kohlbacher [28]. The transcribed interview data was processed by summarizing the key themes to capture the main ideas from the informants. These themes were then used as a basis for further explication of the data. Dimensions of structuring became apparent from the data and the results are presented within those dimensions.

3 Results

3.1 Source

The collected data indicated that one explanation for the large revenue stream variation was the source of revenue, namely the customer or customer segment. As one informant put it: "Typically if they have an organization change then the desired invoicing [method] changes." Thus, a big factor affecting the revenue stream is the customer and their needs. This can be partially due to unbalanced negotiation power between the parties. The dynamic nature of the customer means that the revenue stream is also dynamic in nature. When the source of the stream is dynamic it is reflected in the revenue stream. Within the case firm five different sources of revenue were detected. Four of them were different medium to large companies. The fifth source was a group of small companies. The group was analyzed together as one revenue source, because there were no differences from revenue stream point of view.

All the revenue streams in the current case were negotiated separately on a customer by customer basis as a whole and in some customer accounts different parties were involved in negotiating the managed services and the software development agreements. Actually having to negotiate the pricing in each revenue stream added to the complexity of the sales process. The lack of a price list was mostly due to lack of product management efforts in general. The extent of customer specific negotiations suggests that the customer will have a great impact on the revenue stream making it a differentiating dimension. The customer negotiation intervals also have an effect on the predictability of revenue.

3.2 Reason

While source of revenue was a significant explanatory factor for the variation there were also differences in revenue streams originating from one source and it could be seen that the variation was dependent on the reason for revenue. Reason for revenue can be considered to be the offering item which is the product or service and has in most cases a contractual basis. In the current case 9 different reasons were identified from the interviews. They were: billing manager service, customer care system, order entry system, billing system, keeping the systems running, enterprise resource planning system, system development, consulting/analysis. Additionally the firm offers fixed price delivery projects for new customers before the relationship progresses into so-called operative mode. However, no such delivery was ongoing during the interviews and therefore this aspect was excluded from the study. Focus is on the current customer relationships.

Revenue streams based on different offerings varied in terms of packaging level. The revenue streams whose reason for revenue was system licensing or maintenance service offerings were sold as a complete package. On the other hand those streams whose reason for revenue was system development and customization activities contained various configurations based on customer specific needs.

As mentioned earlier, the source of revenue dimension had a somewhat dynamic nature meaning that the needs change over time. Similarly there was dynamism in the

reason dimension. It was clear that the offerings were not static. There was also a preference towards generating one type of revenue over the other. An informant commented that they try to push more towards the model where the development is less and maintenance is more: "It's changing towards the direction where maintenance portion is growing; it also has the best upside, because tools are automated." This indicates that the reason for revenue -dimension is also dynamic in nature.

3.3 Method

The method of revenue dimension for each stream had differences across streams but similarities as well. Therefore the analysis within this dimension is more involved. In table two the method is described for each revenue stream that is considered unique. In the current case, each revenue source can be considered to originate unique streams compared to other sources, but multiple reasons can exist for the same stream and those reasons can have the same method, so they are combined here into cells depicted in table two.

3.4 Analysis Matrix

It proved useful to present the data in a matrix of reason vs. source where for each cell of the matrix the method of revenue stream was considered. If the reasons were contributing to the same revenue stream they were combined together. This way 11 revenue streams (separate money flows) were identified.

Further looking at these 11 revenue streams and their differences, they could be grouped into four revenue stream types that were considered as unique in the sense that they had a lot of similarities in the method dimension. The four different revenue stream types were given designations A, B, C and D (see table 2).

Table 2. Revenue stream types were grouped based on similar structure of revenue

		Reason for revenue							
		Billing manager service	Customer care system	Order entry system	Billing system	Keeping the systems running	ERP system	System development	Consulting/Analysis
Source of revenue	1	Revenue stream 1: Monthly service fee based on amount of subscriptions..					not offered	Revenue stream 6: Projects fee 8 times per year. Every half year a plan for 6 months of work, and after that invoice the extras. Analysis phase invoiced when leading to development. Unused reserved capacity partially invoiced.	
	2	Revenue stream 2: Monthly maintenance fee based on amount of customers with active subscriptions..					not offered	Revenue stream 7: Development fee 8 times per year. Based on hours but adjusted up or down based on the benefit that the customer would perceive they get, breakdown to analysis, development, etc.	Revenue stream 8: Analysis invoiced full-time and separately.
	3	not offered	not offered	not offered	Revenue stream 3: Fixed usage/license fee invoiced monthly.		not offered	Revenue stream 9: Variable development fee invoiced monthly based on worked hours.	
	4	not offered	not offered	not offered	not offered	Revenue stream 4: Maintenance fee invoiced quarterly in advance. Fixed amount.		Revenue stream 10: Development fee monthly afterwards based on worked hours.	not offered
	5	not offered	not offered	not offered	Revenue stream 5: Maintenance fee invoiced quarterly. Fixed amount.		not offered	Revenue stream 11: Development fee monthly afterward based on worked hours.	not offered

Stream type A

Stream type B

Stream type C

Stream type D

3.5 Revenue Stream Types

Stream type A consists of similarly structured revenue streams one and two. For both of them the method of payment is a fixed fee and a per unit price. For the stream 1 the unit is per active subscription and for stream 2 the unit is per end-customer (customer's customer) who has an active subscription handled by the system. In the interviews it was suggested that due to foreseeable changes in the industry the preferred model from vendor perspective was seen to be per service per customer which would better reflect the cost structure and allow the provider to benefit from the new services they might need to support by the system. In general the benefit of the invoicing tied to the growth of subscriptions was seen in having a shared goal of helping the customer grow, because it means more money for the vendor as well. Informant number five commented that "this is the best model I know".

Revenue stream type B included streams where the name people used for the model was different but the formula was the same, so it can be considered one stream type. The terms were either usage fee, license fee or maintenance fee, but they were all basically a fixed amount invoiced at a regular interval, either monthly or by quarterly, in advance or afterwards. Stream types A and B are basically the same in terms of offering: system usage right, and maintenance service. The terminology is interestingly causing problems. Informant six noted: "Because we charge license fee we have a lot of problems, because they see that they should get monthly development for free." Many people also felt that the future model should be more geared towards per unit based invoicing, because it offers a possibility to move toward value based invoicing away from cost based invoicing. Still, for new customers the downside is increased risks as informant 11 put it: "There is a challenge, because there are not that many of those and for us the cost of hardware doesn't go down. [In case of] minimum monthly payment, the volume can be too low, too much risk." This is one of the reasons why revenue stream type B exists alongside A. It was a safe choice at the initial selling stage.

Revenue stream type C is an interesting one, because it includes a guaranteed minimum purchase. Thus it could be called assured purchase volume and per unit invoicing. The way this is done in practice is that there is a planning session every half a year for the upcoming work which is partially guaranteed work. Informant 1: "Current agreement offers us safety, that we have half a year work at a time. We can invoice 80 percent even if they would order nothing". Otherwise work is invoiced on a per unit price rate where the unit is the amount of worked hours.

Revenue stream type D is a plain per unit invoicing. Compared to stream type C the vendor takes the bigger risk. Pure per unit invoicing was considered easier to sell. The benefits of having an assured purchase volume were seen mainly due to the low transferability of excess capacity between the teams producing the offerings that generate the revenue streams. In the current case this low transferability problem is interestingly solved not by developing the organization but rather creating a revenue stream method that allows it.

Table 3. Contributions of revenue reasons to total revenue from each source

		Reason for revenue							
		Billing manager service	Customer care system	Order entry system	Billing system	Keeping the systems running	ERP system	System development	Consulting/Analysis
Source of revenue	1	70 %					not offered	30 %	
	2	45 %					not offered	45 %	10 %
	3	not offered	not offere	not offere	50 %		not offered	50 %	
	4	not offered	not offere	not offere	not offere	17 %		83 %	not offere
	5	not offered	not offere	not offere	95 %		not offered	5 %	not offere

3.6 Revenue Contributions

There was variation between the percentage contributions of revenue reasons to total revenue from each source. Table 3 summarizes these percentages and shows the differences between how much each revenue reason group contributes to the total income when comparing revenue sources to each other. The variation could be due to the lifecycle of the revenue source and one could guess that a new customer would require more development related activities whereas older customers would only need the service contract. There is initial support for such a conclusion, but the interviews indicated that other reasons like who made the original contract had more effect. Still, the interviews indicated that there was a goal to move away from stream types C and D towards stream types A and B. This was related to the fact that development work is dependent on doing more work: "In the development side the upside will not be very high. It always includes a lot of work." There was an element of unpredictability about future revenue. The fact that the buyer can decide upon buying something or not was seen bad and offering as a packaged service was preferred: "Rather predictability is better, so service fee [is preferred]." It could be said that revenue contributions overall are more likely to move towards the service oriented stream types A and B over time.

In sum, the undertaken analysis approach helped clarify the revenue stream variation within the case firm and gave support for the decision makers' business model understanding. During the interviews one of the informants had commented: "It's hard to tell which revenue stream contributes what. Because it seems the money goes into one bucket." Introducing the revenue stream type analysis can be the first step to alleviate the situation and help the firm in strategic decision making.

4 Discussion

The goal of this paper was to conduct exploratory research and answer the question about the constituents of revenue stream. The study suggests that a revenue stream has three main sub-components which are the source of revenue stream, the reason for the revenue stream and the method of revenue stream.

Table 4. Sub-components of revenue stream

Component	Definition	Examples
Source	The originating source of revenue flow from whom does the money come from.	Specific customer, customer segment, consumer segment.
Reason	The reason(s) why someone is paying the money.	Offering item, service or product, contractual relationship.
Method	The method of how the payment occurs and how it is structured.	Paid every month based on amount of worked hours with a minimum invoicing.

The second question to answer was how the revenue stream can be analyzed within a firm. It has been stated in previous literature that a business can produce one or more revenue streams from each source customer segment [17]. In this paper it has become evident that in a complex b2b setting, one revenue stream can be caused by several reasons of revenue each having different methods of revenue. In addition varying revenue streams can originate from similar sources. In the current b2b case the complexity was such that it was confusing to try to explain it without a clear structure or fit it into a too abstract model. It is suggested that revenue streams should be analyzed so that the method of getting paid is considered for each cell of a two dimensional matrix having two axis: source of the revenue stream and reason for the revenue stream. Only after this kind of analysis can the similarities in method of capturing revenue between the streams warrant a recombination into revenue stream types with similar attributes. Osterwalder [18] uses term stream type very broadly to mean type of economic activity used to generate income. Stream type is also often reduced to just listing examples such as: selling, lending, licensing, transaction cut and advertising [29]. This paper suggests, however, that this simplification is not necessary or even applicable in the current case and a revenue stream type within a firm should be defined based on a comparison of methods of revenue viewed through a source by reason matrix. Thus, it is suggested that a revenue stream type describes the method of revenue for streams originating from a similar revenue source for a similar reason for revenue. Further a full explanation of a revenue model means describing all the revenue stream types used.

Based on the empirical analysis the following hypothesis is suggested for future testing: When analyzing the revenue streams of a business model, it is necessary to analyze them separately based on source (from whom does the revenue originate from?) and reason (on what offering is the invoicing based on?) dimensions. Further it is suggested that analyzing the method of revenue within these "source-reason" cells allows the detection of unique revenue stream types which define the nature of the business model in regards of revenue. This kind of matrix cell representation is suggested to describe the firm's revenue mix much better than for example the revenue mix concept of Stähler [30] which is defined as the sum of all sources of revenue the firm has. It is suggested that revenue mix concept should rather be a description of revenue stream types in all three mentioned dimensions not just the one.

Because of the demonstrated incoherence of revenue aspects relating to business model in the existing literature, a decomposition of the revenue stream concept was attempted. This paper provides support to the usefulness of the concept revenue stream and suggests its applicability also to the analysis of b2b software service businesses. Because only one specific context of b2b software service business was considered, further study should be made in other contexts to compare the findings and investigate the suggested decomposition to enable more general theoretical propositions. Here the context was b2b, because of the case selection, but it might be possible to expand the findings towards b2c in the future.

This paper contributed to the business model research by defining three constituents of a revenue stream and introducing the concept of revenue stream type as a combination of revenue streams with similar method of revenue. For management practitioners a tool was presented for analyzing revenue aspects of the business model. The presented decomposition could be used when investigating for example the profitability of different revenue streams to gain a more fine grained analysis. Managers can use this systematic approach to better understand the business and describe and visualize the revenue streams involved.

The suggestion for future business model research is to promote the money flow i.e. revenue stream as the central concept around which an analysis of a business model should be built upon, because a business by definition has to generate revenue in order to be viable on the long term. Therefore, following the money is a good idea.

References

1. Petrovic, O., Kittl, C., Teksten, R.: Developing business models for ebusiness. Available at SSRN 1658505 (2001)
2. Al-Debei, M.M., Avison, M.: Developing a unified framework of the business model concept. European Journal of Information Systems 19(3), 359–376 (2010)
3. Osterwalder, A., Pigneur, Y.: An e-business model ontology for modeling e-business. In: 15th Bled Electronic Commerce Conference, Bled, pp. 17–19 (2002)
4. Pateli, A., Giaglis, G.: A framework for understanding and analysing e-business models. In: Bled Electronic Commerce Conference, Bled (2003)
5. Mahadevan, B.: Business models for Internet-based e-commerce. California Management Review 42(4), 55–69 (2000)
6. Rajala, R., Rossi, M., Tuunainen, V.K.: A framework for analyzing software business models. In: ECIS, pp. 1614–1627 (2003)
7. Weill, P., Vitale, M.R.: Place to space: Migrating to eBusiness Models. Harvard Business Press (2001)
8. Alt, R., Zimmermann, H.D.: Preface: introduction to special section–business models. Electronic Markets 11(1), 3–9 (2001)
9. Magretta, J.: Why Business Models Matter. Harvard Business Review 80(5), 86–92 (2002)
10. Afuah, A., Tucci, C.L.: Internet Business Models and Strategies: Text and Cases. McGraw-Hill, Boston (2001)
11. Gordjin, J.: Comparing two business model ontologies for designing e-business models and value constellations. Technical report, 18th Bled eConference eIntegration in action (2005)

12. Rédis, J.: The impact of business model characteristics on IT firms' performance. International Journal of Business 14(4), 291–307 (2009)
13. Nenonen, S., Storbacka, K.: Business model design: conceptualizing networked value co-creation. International Journal of Quality and Service Sciences 2(1), 43–59 (2010)
14. Schief, M., Buxmann, P.: Business models in the software industry. In: 2012 45th Hawaii International Conference on System Science (HICSS), pp. 3328–3337. IEEE (2012)
15. Ojala, A., Tyrväinen, P.: Revenue models in cloud computing. In: Prakash, E. (ed.) Proceedings of 5th Computer Games, Multimedia & Allied Technology Conference (CGAT 2012), pp. 114–119. GSTF, Singapore (2012)
16. Zott, C., Amit, R.H.: Designing your future business model: an activity system perspective. Social Science Research Network Working Paper Series (2009)
17. Osterwalder, A., Pigneur, Y.: Business model generation–a handbook for visionaries, game changers, and challengers. Wiley, New York (2010)
18. Osterwalder, A.: The business model ontology: A proposition in a design science approach. Institut d'Informatique et Organisation. Lausanne, Switzerland, University of Lausanne, Ecole des Hautes Etudes Commerciales HEC, 173 (2004)
19. Shafer, S.M., Smith, H.J., Linder, J.C.: The power of business models. Business Horizons 48(3), 199–207 (2005)
20. Linder, J., Cantrell, S.: Carved in water: Changing business models fluidly. Accenture Institute for strategic change (2000)
21. Rajala, R., Rossi, M., Tuunainen, V.K., Vihinen, J.: Revenue Logics of Mobile Entertainment Software-Observations from Companies Producing Mobile Games. JTAER 2(2), 34–47 (2007)
22. Giesen, E., Berman, S.J., Bell, R., Blitz, A.: Three ways to successfully innovate your business model. Strategy & Leadership 35(6), 27–33 (2007)
23. Chesbrough, H.: Business model innovation: it's not just about technology anymore. Strategy & Leadership 35(6), 12–17 (2007)
24. Yin, R.K.: Case study research: Design and methods, vol. 5. Sage (2003)
25. Yin, R.K.: Case study research: Design and methods, 2nd edn. Applied social research method series, vol. 5. Sage Publications (1994)
26. Eisenhardt, K.M.: Building theories from case study research. Academy of Management Review 14(4), 532–550 (1989)
27. Luoma, E., Frank, L., Pulkkinen, M.: Overview of telecom operator software market. In: Vertical Software Industry Evolution, pp. 35–42. Physica-Verlag HD (2009)
28. Kohlbacher, F.: The use of qualitative content analysis in case study research. In: Forum Qualitative Sozialforschung/Forum: Qualitative Social Research, vol. 7(1) (2006)
29. Scheithauer, G., Wirtz, G.: Business modeling for service descriptions: a meta model and a UML profile. In: Proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling, vol. 110, pp. 79–88. Australian Computer Society, Inc. (2010)
30. Stähler, P.: Business models as a unit of analysis for strategizing (2002), `http://www.scribd.com/doc/34770740/Business-Models-as-a-unit-of-Analysis-for-Strategizing` (referenced on January 7, 2013)

Defining the Process of Acquiring Product Software Firms

Jasper Schenkhuizen, Robert van Langerak,
Slinger Jansen, and Karl Michael Popp

Utrecht University, Utrecht, The Netherlands
(jasper.schenkhuizen,rvlangerak,slingerroijackers,karlpopp)@gmail.com

Abstract. Product software companies increasingly seek expansion by means of acquiring software products. For product software firms, the process of an acquisition is complex and challenging because acquisitions require complex processes, business risk, and life-changing decisions. The determinants that influence such acquisition decisions are rarely investigated. Prior research has focused on software acquisitions, but has not focused on software acquisition determinants during an acquisition process. In this study, the product software acquisition process has been defined and the determinants have been identified. Experts evaluated and assessed the acquisition determinants and the acquisition process, in order to find a critical determinant for each respective phase. Finally, a model is presented in which the most critical determinants are presented in the different phases of the acquisition process. The results provide an exploratory set of guidelines that help managers at product software companies through the complex processes of acquisitions in the product software industry.

Keywords: product software industry, software acquisitions, software acquisition determinants, portfolio extension, product software acquisition process.

1 Introduction

Mergers and acquisitions characterize the software industry. It is an industry that is consolidating continuously. Advantages of economies of scale, combinatorial sales, and strategic alignment of niche players in ecosystems of large software companies are some of the drivers of this consolidation. In this context the term acquisition refers to the situation where a company buys another company or a set of software product assets of a company, not the purchase of software products from software suppliers for operational use. Sometimes a company is built around one software product. During the acquisition of such a software product this leads to the takeover of the entire company. In general this paper focuses on the acquisition of a software product as an asset. Acquisitions are complex processes [7,12,19], where attention should be paid to different aspects. In the product software industry, attention should be paid to product software related

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 100–114, 2014.

aspects besides financial, economical, political and legal issues. Software acquisition determinants are factors that are used by decision makers to decide whether an acquisition will proceed. These factors are rarely researched, although they are the determining factors for the process of software acquisition and for the success of such a project. On average, in the worldwide economy, about 20% of the mergers and acquisitions succeed [6], however, to our best knowledge, the ratio for the software industry is not yet documented. There is little scientific literature available about the process of acquisitions in the product software industry, which is remarkable because the industry is flourishing and the number of mergers and acquisitions is increasing: in 2012, the number of announced software M&A deals worldwide was approximately 1900, while in 2009 the amount was 1500 [22].

This study attempts to fill the gap in the body of knowledge concerning both the process of acquisitions in the product software industry, and the software aspects that are related to an acquisition. There are few studies focusing on acquiring software products from software vendors and on software quality assessment, but not specifically on product software acquisitions. The immature research that is done regarding product software acquisitions can therefore be identified as a gap in the body of knowledge, which this study starts to fill by defining the product software acquisition process, and identifying the most critical product software acquisition determinants that influence decisions during the complex process of software acquisitions. The results of this study can be used to help improve the understanding of acquisitions in the product software business.

This paper is structured as follows: we continue the discussion of related literature in Section 2 and highlight how and why the product software industry differs from other industries. In Section 3 the research method is described, including a detailed discussion of the interview process with eight European and American acquisition experts. In Section 4 the product software acquisition process is defined, based upon interviews and literature. The product software acquisition process can be considered as one of the key contributions of this paper. Section 5 highlights the product software acquisition determinants and the weights that they were assigned by the experts in the different phases of the acquisition process. We continue Section 6 with a discussion of the results and identify the weaknesses of the research. Finally, in Section 7 we conclude that the process and weighted determinants contribute to the body of knowledge on product software acquisition and expect that acquirers of software companies are helped by the insights provided in this paper.

2 Related Literature

The collection of scientific work on mergers and acquisitions is diverse, but the specific focus on software acquisitions - or mergers has not yet received a lot of attention. Best practices and standards from the ISO/IEC and the IEEE discuss software acquisition: the ISO/IEC 25040 [10] standard provides approaches for

measurement and evaluation of software product quality. It is appropriate for acquirers, but does not give a clear overview of the process and the determinants. Another standard, developed in 1993, is the Recommended Practice for Software Acquisition, IEEE Standard 1062 [8]. It presents a set of recommended practices which can be applied on different types of software. The focus is on the software itself, not on the process and its associated determinants. An updated version of the IEEE Standard 1062 [8] is the IEEE Project P1062 [9]. It is a project containing best practices that help and support organizations to make a selection, evaluation and eventually help accepting supplier software for operational use. Furthermore, Nelson, Richmond and Seidman [14] developed a decision framework for a two-dimensional problem that they call the software acquisition problem. Although these related sources might be useful during the acquisition of a product software firm or software assets of a firm, they are too specifically focused on the acquisition of software packages from vendors for operational use, instead of actually acquiring specific software assets. Related work that lies more in line with this study is coming from The Software Improvement Group (SIG) [20], which conducted research on due diligence in the software industry. Nonetheless, the SIG does not cover the acquisition process. Related work that does relates to multiple aspects of this study is work of Popp [17]. Popp identified a software acquisition process, which is used to develop the acquisition process for this paper. Work of Popp [17] was found very useful and discusses software due diligence very extensively.

The product software industry differs from other traditional industries in various ways, which results in fairly complex acquisitions in the product software industry. Related work from Popp [17] shows that software ecosystems are self-organizing, something that makes predictions about the software industry complicated [17]. Software companies are often part of one or more software ecosystems, resulting in difficult predictions about the acquisition environment. Furthermore, when a software product is finished, the development of hundred or thousand of the same products costs almost the same, in other words *"the cost of duplicating a software product is nearly zero"* [21]. There is no other business that has a gross profit margin of 99 percent. Traditional manufacturing firms, where products are tangible, duplicating products is costly. The low cost of duplicating a software product results in a difficult determination of a software firms value, therefore making software acquisitions complex. Furthermore, according to Beizer [1] software is complex, hard to build and has no physical barriers. He notes that software strategies are often based upon assumptions that software will behave sensibly. Since the behavior of software is not comparable with physical objects, software is different from physical objects, and therefore the product software industry differs from other regular industries. Moreover, Popp [18] notes that the product software business has a high importance of workforce quality due to low automation in production. It originates from a statement that Nowak and Grantham [15] make: *The knowledge encapsulated in software will increasingly define the economic value of the intellectual capital it represents.* Software is built from human capital, which requires knowledge and

communication, thus there is less need of natural resources and physical labor. This makes the production process in the software industry difficult to automate and results in a high importance of workforce quality, therefore differentiating itself from several other industries. Furthermore, in most cases the vendor of product software retains ownership of the software product when the product is sold [25], opposed to for example manufacturing industries. Also, software often consists of multiple modules, developed by multiple firms, resulting in ownership issues. A consequence is a strong role of intellectual property rights in the product software business [18]. In addition, the diverse range of business, revenue and delivery models raise challenges in the area of finance and taxation.

All the above mentioned differentiators result in the need for a restructured view on acquisitions in the software industry, since these differentiators combined give a good view on how the product software industry differs from other industries and how some aspects make the industry quite complex.

3 Method

The empirical part of this study aimed at prioritizing critical determinants in the different phases of the product software acquisition process, which is described in Section 4. In order to do so, the 6-phase design science method has been used [3]. This method consists of a 6-phase process, in which a test artifact is created, evaluated and eventually finalized into a model. A prerequisite for collecting data for this study was a clear definition of the product software acquisition process and an initial list of critical determinants. A literature study was conducted to define the steps and activities of the product software acquisition process. The initial list of critical determinants has been based upon literature as well as on preliminary expert interviews. These critical determinants are particularly related to software acquisitions with portfolio extension associated incentives. Acquisitions with market consolidation related incentives probably have other determinants, and are outside the scope of this study. To evaluate the correctness, accuracy, and validity of the determinants and the process, an expert evaluation in the form of semi-structured interviews was performed. The main part of the study consisted of a cross-evaluation of the model by eight experts.

3.1 Experts

In total, eight experts have contributed to this study by participating in semi-structured interviews. We defined an expert as someone who had been involved in several software acquisitions, in which the expert fulfilled an advisory or decisive role. These experts have been carefully selected based upon experiential and educational relevance: they were working or had worked at different sizes of software business related companies and institutions, and most of them have been graduated in the field of finance, economics or a technology related study. All together, the experts had been involved in more than 250 software related acquisitions, and in total have more than 50 years of experience in the field

of mergers and acquisitions. More information about each expert has been presented in Table 1. The experts fulfilled several roles during these acquisitions, from Lead Advisor (LA) to Leading Due Diligence (LDD), and from Lead Director (LD) to activities regarding Post Merger Integration (PMI). They were involved in deals with for example Sybase, Hybris, Technidata, Twinfield, Addison, FRS Global, Google, Microsoft, Intel, Symnatec. The name on the bottom row is confidential and therefore a generic descriptive term (Leading Electronical Equipment Provider) expressed in italics is used. Other confidential information is indicated with a hyphen (-).

Table 1. The current function and company of the involved experts in this study

Company	**Current function**	**Role during M&A's**	**# of acquisitions**
SAP	Senior Director M&A	LDD & PMI	25
SoftwareAG	Director M&A	LA	-
ISVWorld	Founder & CEO	LA	80
Accountview	CEO	LDD	6
Corum Group	President	LA	100+
Atego Group	President Strategy	LD	30
Arma Partners	Associate	LDD	8
LEEP	Senior advisor M&A	LA & PMI	5

3.2 Materials

In the phase prior to the data collection, Microsoft Excel has been used to share the initial list of determinants with the experts. The interviews used for data collection have been recorded with a smartphone audio recorder and a program called Skype Call Recorder, which has the capability to record both the video and audio of a Skype call. During the interviews, the model was presented to the expert in a spreadsheet of the on-line collaboration service from Google, called Google Drive. This particular software offered the possibility for us to work in the same document as the expert simultaneously. Seven of the interviews have been performed via Skype, and one interview has been performed via a landline telephone.

3.3 Protocol

The evaluation of the correctness of the initial list of determinants was performed by two experts, who received the Excel-spreadsheet via email. When the experts finished their evaluation of the process and the list of determinants, they returned

the Excel-file via e-mail. During a Skype-conversation, the comments were discussed and processed. Based on these preliminary results a selection was made of 20 determinants which were used in data-collection-phase. The data-collection-phase of the study involved the participation of eight experts in interviews that lasted for 50 minutes on average, with the shortest interview being around 30 minutes and the longest interviews about 90 minutes. The interviews consisted of three parts: firstly, the expert received an introduction about the study and the purpose of the interview. Then some questions were asked to create a context of the experts experience and education. Thirdly, participants received an on-line spreadsheet, in which the test model was depicted and described. Before starting the actual data collection, the on-line spreadsheet was exemplified to the expert. The participant was asked if he was missing any relevant determinants and understood the present determinants. In addition, the expert was asked if he understood and agreed upon the defined acquisition process depicted in the spreadsheet. Eight experts were requested to rate the determinants for each phase of the acquisition process, from 0 to 10, whereby 0 would refer to not taken in consideration and 10 would refer to highly critical. The experts were asked to think aloud while filling in the spreadsheet, and to argument and motivate their choices. The conversations have been recorded and anonymously transcribed for later analysis. When experts were missing certain determinants in their opinion, there was the possibility to add them and rank them accordingly. If such a change was made to the test model, afterwards the determinant was taken into consideration and possibly added to the test model. This has led to an incremental and iterative growth of the model. When a determinant was added after an expert had participated in the study, the expert was requested to evaluate and rate the new determinants in the same on-line spreadsheet. This was done to assure that all experts reviewed all determinants, thereby cross-evaluating the model. The readability of the model was improved by splitting the determinants in two groups: software product properties and business environment properties.

4 Product Software Acquisitions

An incentive for acquisitions is growth [4], because at the end of the day a company only wants to increase its value [2,13]. Synergistic benefits are incentives as well [4,13,2,24]. Other incentives are diversification of a companys products [4] and expanding the product portfolio [17]. An incentive that belongs to a more long-term strategy is to seek for early winners. Potential successful companies that deliver products with possibly high values are called early winners [11]. When an acquirer recognizes potential growth, long before others do, the acquirer can make an acquisition at a relatively reasonable price and gain advantage of the acquisition later [4,24,11]. Another incentive is to buy a company for gaining knowhow advantages and skills. This includes acquiring technologies faster and at lower cost than they can be built by the acquirer itself. It results in low development costs, the risk of building a bad solution will be avoided and the knowhow of the R&D department is improved. In addition, there will be a

faster innovation of products by continuously buying new emerging ideas [4,11]. One or more of these incentives will eventually lead a company to identifying possible acquisition targets, which is the first phase of the acquisition process. This process will be discussed in the next paragraph. As Gomes et al. [5] describe, there is no clear defined consensus of the phases of a merger & acquisition process. This is the result of the possibility to perform different aspects of an acquisition simultaneously [5].

Because phases can overlap, Gomes et al. [5] eventually try to sketch three separated phases: pre-merger, ownership transfer, and post-merger. In order to define a new software acquisition process, several kinds of acquisition processes that are described in literature [16,23,17] have been combined to create an product software acquisition process that we think gives the best representation of an actual acquisition. The process is presented in Figure 1.

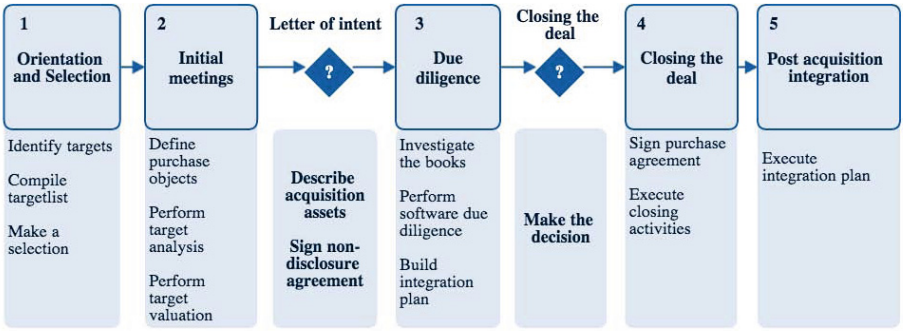

Fig. 1. The product software acquisition process, consisting of five phases, identified from literature and expert evaluation

Phase 1 of the acquisition consists of identifying possible targets, compiling a target list and making a selection. An analysis of alternatives should be made by the acquirer. When a selection has been made, the acquirer should approach the possible target and start initial meetings. During phase 2, the various objects of the purchase will be defined based upon several initial meetings that have taken place. The acquirer has to make a target analysis and do a target valuation, activities that can be performed simultaneously. If the negotiations between the firms are progressing well and the acquirer is satisfied with the result of the target analysis during phase 2, a letter of intent has to be sent to the target. In this letter the objects of the acquisition are further described, including all the contingencies associated with those items [16]. In this letter of intent the two firms agree on a non-disclosure agreement and confirm that they proceed to the following phase of the acquisition process. In phase 3 due diligence activities will take place. Popp [17] notes that the goal of due diligence activities is the ability for the acquirer to make an informed decision about whether to continue.

Besides normal due diligence, the Software Improvement Group [20] states that *software due diligence provides insight into the costs and risks that impact the future of the software investment*, and therefore it indicates important issues when doing software specific due diligence. These issues can be found in documents of the Software Improvement Group [20] and need to be considered when going through the due diligence phase. Furthermore, Popp [17] states that intellectual property due diligence is an important part of software due diligence and needs to be done to *safeguard the existing business of the target, future business and to safeguard that the target really owns their software products.* Paulson [16] and Popp [17] both state that constructing an integration plan during due diligence is crucial for the success of an acquisition in the software business. When the due diligence process is finished the acquirer must decide to continue or stop the acquisition process. This is an important go/no go decision, as a negative outcome ends the whole acquisition. By signing the definitive agreement in phase 4, the deal will be closed [16,17]. This agreement contains several terms and conditions that the buyer and seller will eventually have to agree on. Subsequently, the acquisition price must be defined [16,23]. Both parties need to agree on the integration plan and execute closing activities. The closing conditions must be met before the deal can be closed. When both parties agree on the agreement and the deal is closed, the post acquisition stage is entered. During the post acquisition stage phase 5 - the integration plan is further executed to make effective use of the acquired technology. Depending on the incentive of the acquirer the acquired company is now adapted and aligned to the needs of the acquiring company. This is done by combining or adapting different software structures, solving managerial and cultural issues, directing the orientation and interaction of new employees, implementing procedures and techniques, and arranging any remaining legal problems [23]. Paulson [16] mentions that underestimating the pitfalls in this stage can lead to making serious mistakes.

Acquisition Determinants. The main deliverable of this study is a model that describes the most critical determinants of a software product during the different phases of an acquisition process. As there currently is no definition of product software acquisition determinants, we defined product software acquisition determinants as: *Software product and software business environment related characteristics that exert influence on the decisions that are being made regarding the takeover of product software assets or firms.* Risk factors of an acquisition might incorrectly be considered as product software acquisition determinants, but are not incorporated in this definition. Risks are not a driving factor in acquisitions; rather the outlook to get rid of some risks might be in some case. The initial list of critical determinants has been based upon a literature study as well as on expert knowledge, as described in the method part of this paper. The final list of determinants is constructed as follows:

– **Product Technology:** Describes the technology that is used in a software product, including platforms, standards and operating systems.

- **Source Code Quality:** The quality of the source code, which substantially determines the performance of a software product.
- **Total R&D Investment:** Refers to the relative ratio between research and development investments and the products current value.
- **Business Model:** Describes the strategies and models with which a company attempts to create value.
- **License Model:** The methods and strategies that a company uses for licensing.
- **Profit Margin:** Describe the ratio of profitability, which gives insight in which of the revenues turn into profits.
- **Payment Model:** A companys payment model determines when and what customers pay for the service or product.
- **Key Cost Drivers:** The cost drivers that use most of the companys resources. They give insight in the financial situation of a company or a product.
- **Asking Price:** The initial negotiation price the target firm asks is the asking price.
- **Percentage of Recurring Revenue:** Refers to the revenue that is recurring in a financial year, compared to the total revenue. Recurring revenue often is steady revenue.
- **Revenue Synergies:** Revenue synergies describe the increase of revenue of a company as a result of combining two or more businesses.
- **Cost Synergies:** Describe the extent to which a company can eliminate costs throughout the organization as a result of combining two or more businesses.
- **Market Share:** Describes the percentage of an industrys total sales that is collected by a software firm in a certain time frame. Software firms can achieve a high market share, without actually provide a large part of the users in the industry with their software. If the total of an industrys sales was very low, a high market share does not imply a large user base.
- **Product's Potential Customers:** Describes the potential customer group of the product that is to be acquired. The potential customers can deliver critical information about the potential results of the product.
- **Installed Base:** Refers to the share of customers within a certain industry that are currently owning or actively running the product. A high market share does not imply a large installed base.
- **Intellectual Property Rights:** Intellectual property rights allow creators to protect their ideas and creations.
- **Cost of Mandatory Future Upgrades:** Refers to the projected expenses of future maintenance.
- **Portfolio Fit:** Describes whether the software product that is to be acquired fits in the current portfolio, to a technological and strategic extent.
- **Available Alternatives:** Refers to the existence of alternatives for a particular target product.
- **Key Employee Retention:** Key employees have extensive knowledge of the software and this knowledge about the product makes them valuable

for the acquirer. Retention of employees makes updating and managing the product more efficient.
- **Company Culture:** Describes the conflicting company cultures. Race, hierarchy, beliefs and design culture are examples of conflict topics.
- **Employee Integration:** Describes the difficulty and possibility of the integration of employees into the acquiring company.
- **Cooperation Partners:** Describes the extent to which the target has outsourced the development activities and other processes.
- **Support Model:** Contains the existing collection of support activities the target company currently delivers, to keep the product running as advertised.
- **Services:** Describes the activities of the target company that are performed on top of support activities. Supplementary courses and training programmes are examples of services.

To avoid ambiguity regarding the meaning of a determinant, each factor has been described in the abovementioned list. Each determinant individually exerts influence on the decision-making during an acquisition, and therefore each determinant should be analyzed, guided by the description in the list above.

5 Results

In Table 2 the results of the interviews have been combined. The table shows all determinants, including the following determinants that have been added during the data collection: percentage of recurring revenue, revenue synergies, cost synergies, company culture and employee integration. As mentioned in the method section, all determinants have been cross-evaluated by all experts to ensure the validity of the results. The determinants have been weighted by importance from 0 to 10. Each weight in Table 2 is the average weight of all weights that have been given for that determinant in the associated phase. As stated, the numbers in the cells are based upon the average of the ratings that the experts gave to each determinant, for each phase.

In the first phase Business Model was rated an 8,63, thereby being the highest weighted determinant in its phase. Asking Price received an average weight of 8,13 in the second phase. Intellectual Property Rights was rated a 9,00 in the third phase, and the fourth phase had a relative high weight for Key Employee Retention with 7,38. An average weight of 8,88 was given to Revenue Synergies and Employee Integration in the fifth phase. Table 3 presents the answer to the incentive of this study, by indicating the most critical software acquisition determinants of each phase of the acquisition process.

Business Model was perceived as most critical determinant of phase 1, as many experts noted that it is counterproductive to transform an on-premise based business model to a SaaS business or the other way around. Therefore, this determinant might have a large influence on the process of selection in the first phase. The high weight of Portfolio Fit in the first phase suggests that it is a critical determinant during the selection of potential targets. Experts noted

Table 2. The weights of product software acquisition determinants in each phase, given by eight experts

Phase	**1**	**2**	**3**	**4**	**5**
Software product properties					
Product Technology	6.75	7.00	7.38	2.63	5.63
Source Code Quality	1.63	2.50	8.25	4.00	4.88
Total R&D Investment	4.38	5.13	5.63	2.63	3.13
Business Model	8.63	6.38	6.25	1.75	5.88
License Model	3.00	3.75	3.88	1.63	3.88
Profit Margin	4.88	4.13	6.25	2.75	4.63
Payment Model	1.63	3.25	3.25	1.25	4.13
Key Cost Drivers	1.88	3.50	6.88	2.13	5.63
Asking Price	4.75	8.13	3.88	5.38	1.38
Percentage of Recurring Revenue	6.25	6.50	7.13	4.00	5.13
Revenue Synergies	3.63	6.63	6.25	4.63	8.88
Cost Synergies	2.25	4.00	5.13	4.50	7.38
Market Share	6.75	5.38	3.63	2.50	4.00
Product's Potential Customers	7.38	5.13	5.75	2.50	5.13
Installed Base	7.13	5.13	6.25	3.75	4.88
Intellectual Property Rights	6.25	5.75	9.00	7.25	4.88
Cost of Mandatory Future Upgrades	2.13	3.75	6.25	2.50	5.00
Business environment properties					
Portfolio Fit	7.50	7.50	4.25	3.00	5.88
Available Alternatives	7.38	4.13	2.50	3.25	0.13
Key Employee Retention	1.50	3.25	6.38	7.38	8.13
Company Culture	3.00	5.00	6.50	3.75	7.75
Employee Integration	0.75	2.63	4.63	2.88	8.88
Cooperation Partners	2.50	4.50	5.38	3.25	5.38
Support Model	2.38	3.63	3.50	1.75	5.63
Services	3.88	2.88	4.50	1.63	4.75

that firms that want to acquire search for companies that have products that match the product portfolio, and often base decisions upon this determinant. Products Potential Customers also received a high weight during the first phase, which suggests the potential customers of a target company or product are of importance when selecting a target. Due to the fact that the acquirer can sell already existing products to a network of potential customers, distribution synergies are achieved. Surprisingly, the determinant with the highest weight in the phase of initial meetings was Asking Price, what might seem early on in the process. Experts clarified the high weight of the determinant: if the target firm has an initial asking price that is too high in the eyes of the acquirer, then the process will probably stop after one meeting. We find a lower weight of Asking Price in the due diligence phase, but an increased weight in the closing phase. Negotiations about the price might push the weight of this determinant up in the closing phase. Product Technology also received a high weight in the second phase. If an acquirer has built every product on Java, and the target company

Table 3. The most critical software acquisition determinants

	#1	#2	#3
Phase 1	Business Model (8.63)	Portfolio Fit (7.50)	Product's Potential Customers (7.38) & Available Alternatives (7.38)
Phase 2	Asking Price (8.13)	Portfolio Fit (7.50)	Product Technology (7.00)
Phase 3	Intellectual Property Rights (9.00)	Source Code Quality (8.25)	Product Technology (7.38)
Phase 4	Key Employee Retention (7.38)	Intellectual Property Right (7.25)	Asking Price (5.38)
Phase 5	Revenue Synergies (8.88) & Employee Integration (8.88)	Key Employee Retention (8.13)	Company Culture (7.75)

has built everything on .NET, then that could cause severe integration problems and costs. Experts referred to this problem as oil and water, not only being a product challenge in terms of integrating things but also a cultural challenge to the extent that attitudes towards standards and platforms may differ between employees of both companies. The due diligence phase did not yield surprising results, as highest-weight determinants are related to software quality and intellectual property. One expert surprisingly mentioned that Source Code Quality was not critical since the cost of deeply investigating the source code would not outweigh the added weight of the acquisition. Opposed to this expert, the other experts mentioned that software due diligence is a critical activity of the acquisition process, in which the source code quality deserves a main focus. The findings for the fourth phase support the related work from Nowak and Grantham [15] who state that software is build from knowledge, which requires human capital and communication. They argue that human capital is valuable in software. The determinant Key Employee Retention received a high rating in the fourth phase of the acquisition process, most likely since employees know all the ins and outs of a product or company, making them valuable assets of a company and therefore need to be retained as long as possible, according to the experts. Retention of key employees is done in the closing phase, since contracts are signed in this phase. As expected, Asking Price and Intellectual Property Rights are in the top three of critical determinants for this phase. The last phase contains two determinants with the highest identical weight: Employee Integration and Revenue Synergies. Experts mentioned integrating the retained employees is even more important than only retaining them, since retained employees that are not being integrated well, will leave the company sooner. According to the experts, Company Culture plays a significant role when trying to integrate employees into the acquirers firm, and therefore received a high weight in the integration phase.

6 Discussion

Firstly, the possibility exists that economical changes in this industry can cause the acquisition process to change, and technological changes can cause a shift in perceived importance of certain software acquisition determinants. This might lead to different results when conducting the same study again. These influencing factors are circumstances beyond our control. Secondly, the use of different experts plays an important role to improve the reliability of this study and attempting to avoid skewed results. Further, in order to preserve validity, each determinant was clearly defined in a note that was attached to each item in the list. Biases might have been avoided using experts from different industries, with different backgrounds and educations. However, experts in the domain of acquisitions in the product software industry are scarce and a larger sample group might have yielded enough data to perform statistical analysis/quantitative research. A limitation might have been caused by possible pressure that the expert perceived during the interviews. The interviewers could have influenced the attitude and the behavior of experts, as experts could have had the idea that they had to conform to certain expectations. In the attempt to eliminate this limitation, the interviewees were told the data was completely anonymous, in order not to build the pressure. In addition, experts were told to think aloud while evaluating the model, and were not guided by the interviewers when filling in the weights. Despite these efforts to overcome this limitation, it is not entirely negligible, as human beings cannot be entirely objective. Due to the scarceness of experts on this topic in the Netherlands, experts from Germany, England and America participated in this study. Some of the experts had English as a first language, while some of the experts had English as second language. These differences in languages might result in limitations regarding the interpretation of the English and German language.

7 Conclusion

As stated in the introduction of this paper, mergers and acquisitions are very complex activities, where attention should be paid to a variety of different aspects of a company. The findings of this study suggest that software acquisition determinants play a key role during acquisitions in the product software industry. This paper presents the product software acquisition process and the determinants that play a role in the different phases of software acquisition. We expect that acquirers of software companies are helped by the insights provided in this paper, and that the results provide an exploratory set of guidelines that help managers at product software companies through the complex processes of acquisitions in the product software industry. The product software acquisition process and the accompanying determinants are an interesting topic of study, which offers a variety of research challenges. Interesting results could probably be discovered in a study that examines the relation between the size of the company where an expert works, and the rating that is given to each determinant. In

addition, a study that explores statistical correlations or dependencies between determinants might as well be very valuable. It might also be interesting to perform a study on what the results would be if this research focused on market consolidation based acquisitions, as well as what the differences between both types of acquisitions would be.

References

1. Beizer, B.: Software is different. Annals of Software Engineering 10(1-4), 293–310 (2000) [1022-7091]
2. Ceausescu, A.: Merger and Acquisition-A Strategic Option for Companies. Annals of the University of Petrosani, Economics 8(1), 59–64 (2008)
3. Ellis, T.J., Levy, Y.: A guide for novice researchers: Design and development research methods. In: Proceedings of Informing Science & IT Education Conference, InSITE (2010)
4. Gaughan, P.: Mergers, Acquisitions and corporate restructurings. Wiley & Sons, Inc., Hoboken (2007)
5. Gomes, E., Angwin, D.N., Weber, Y., Tarba, S.Y.: Critical Success Factors through the Mergers and Acquisitions Process: Revealing Pre-and Post-M&A Connections for Improved Performance. Thunderbird International Business Review, 55(1), 13-35 (2013)
6. Grubb, T.M., Lamb, R.B.: Capitalize on merger chaos: six ways to profit from your competitors' consolidation and your own. New York Press, 9-10, 12-14 (2000)
7. Hayward, M.L.A.: When Do Firms Learn from Their Acquisition Experience? Evidence from 1990-1995. Strategic Management Journal 23(1), 21–39 (2002)
8. IEEE Standards Association.: IEEE Standard 1062. Recommended Practice for Software Acquisition (1993)
9. IEEE Standards Association.: IEEE Project P1062. Recommended Practice for Software Acquisition. (2003)
10. International Organization for Standardization.: ISO/IEC 25040. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Evaluation Process (2011)
11. Koller, T., Goedhart, M., Wessels, D.: The Five Types of Successful Acquisitions. In: McKinsey On Finance, Perspectives on Corporate Finance and Strategy. McKinsey&Company, number 36 (2010); excerpted from: Koller, T., Goedhart, M., Wessels, D.: Valuation: Measuring and Managing the Value of Companies, 5th edn. John Wiley & Sons, Hoboken (August 2010)
12. Larsson, R., Finkelstein, S.: Integrating Strategic, Organizational, and Human Resource Perspectives on Mergers and Acquisitions: A Case Survey of Synergy Realization. Organization Science10(1), pp. 1–26 (1999)
13. Nakamura, H.R.: Motives, partner selection and productivity effects of M&As: the pattern of Japanese mergers and acquisitions. Doctoral dissertation, Stockholm School of Economics (2005)
14. Nelson, P., Richmond, W., & Seidmann, A.: Two dimensions of software acquisition. Communications of the ACM 39(7), 29–35(1996)
15. Nowak, M.J., Grantham, C.E.: The virtual incubator: managing human capital in the software industry. Research Policy 29(2), 125–134 (2000)
16. Paulson, E., & Huber, C.: The technology M&A guidebook. 1st edn. Wiley (2001)

17. Popp, K.M.: Mergers and Acquisitions in the Software Business. Books On Demand, Norderstedt (2013)
18. Mergers and acquisitions in the software industry, http://drkarlpopp.com/MergersandAcquisitionsinthesoftwareindustry.html
19. Shanley, M.T., Correa, M.E.: Agreement between Top Management Teams and Expectations for Post Acquisition Performance. Strategic Management Journal 13(4), 245–266 (1992)
20. Software Improvement Group, http://www.sig.eu/en/Services/Software_Due_Diligence
21. Suarez, F.F., Cusumano, M.A., Kahl, S.J.: Services and the business models of product firms: An empirical analysis of the software industry. Management Science 59(2), 420–435 (2013)
22. Thomson Financial, Institute of Mergers, Acquisitions and Alliances (IMAA), http://www.imaa-institute.org/statistics-mergers-acquisitions.html
23. Very, P., Schweiger, D.M.: The acquisition process as a learning process: Evidence from a study of critical problems and solutions in domestic and cross-border deals. Journal of World Business, 36(1), 11–31 (2001)
24. Vos, E., Kelleher, B.: Mergers and takeovers: a memetic approach. Journal of Memetics Evolutionary models of information Transmission 5(2) (2001)
25. Xu, L., Brinkkemper, S.: Concepts of product software. European Journal of Information Systems16(5), 531–541 (2007)

Productization of an IT Service Firm

Kadri Guvendiren, Sjaak Brinkkemper, and Slinger Jansen

Dept. of, Information and Computing Sciences,
Utrecht University, Princetonplein 5, 3508 TB Utrecht, The Netherlands
{K.Guvendiren,S.Brinkkemper,Slinger.Jansen}@uu.nl

Abstract. Two types of businesses dominate the landscape of the IT industry: service businesses that develop tailor-made software based on customer specific needs on the one hand, and software businesses that develop standard software products based on market needs on the other hand. The so-called productization process enables software companies to perform a business transformation from customer specific service-driven to a product business. This research aims to evaluate to what extent the proposed productization process is applicable in a service-oriented company using seven theory-testing case studies within the context of one IT service firm. The results indicate that the productization process cannot be applied to its full extent, since most of the product-driven processes are not mature, which is largely caused by a lack of knowledge about the productization process in service firms.

Keywords: Productization, Software product management, product software, business transformation, custom versus standard software development.

1 Introduction

Software Product Management (SPM) is defined as the discipline and business process, which governs a product from its inception to the market or customer delivery and service in order to generate the biggest possible value to the business [5]. The product manager executes this important task. Bekkers, Weerd, Spruit and Brinkkemper [3] developed a competence model for SPM, which includes the relevant focus areas such as Portfolio management, Product planning, Release planning and Requirements management. These focus areas aid product managers in managing or improving their SPM practices in the organization. Based on the focus areas in the SPM, Van de Weerd, Bekkers, and Brinkkemper [18] developed a maturity matrix, which will guide further development of methodical support in SPM. The matrix is used to assess an organization's current software product management capabilities and offer local, incremental improvements to the product managers.

We distinguish two types of software: Customized and Standard software. Customized software is software that is tailored to the needs of one specific customer with the purpose to satisfy that customer, whereas standard software is software that is designed based on the needs of a specific market. Some software producing organizations follow the customer-oriented approach in their strategy while others

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 115–131, 2014.

choose to follow a market-oriented approach. Both approaches have their own advantages and disadvantages [15]. We term software producing organizations that develop standard software *Product Software Companies*, whereas software producing organizations that develop custom solutions are termed *IT Service Firms*.

Artz et al. [1] found that several software producing organizations that develop customer specific software have identified a need to transform to developing and selling standard product software, analogous to mechanical engineering, where serial offerings are seen as the holy grail of scaling up mechanical engineering services. In their research, we term the transformation process from custom software projects to software products the *Productization Process*. The productization process, as introduced by Artz et al., consists of six stages as shown in figure 1.

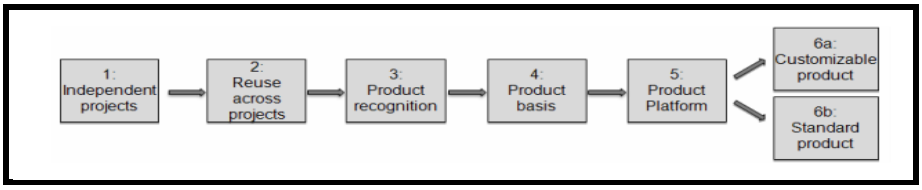

Fig. 1. Productization process

The productization process assists organizations in becoming a product software company, in the case where they are not born as product firms [25]. Artz et al [1] evaluated the process, using methods such as expert reviews and a case study at a firm that has gone through all stages except 6b. The authors recognize that more thorough validation is necessary to prove the applicability of the productization process. This paper is a follow-up of the research of Artz et al. [1]. We aim to further validate the process in an IT service-oriented company, where seven products are assessed on their maturity from a market-driven perspective. This leads to the following research question:

To what extent is the productization process applicable in an IT service firm when transforming from developing customer specific software solutions to standard product software for a market?

In the following section, we first describe the approach taken to validating the productization approach of Artz et al. [1]. We continue in Section 3 with a discussion of Artz's approach and filled in some of the details that are specific to IT service firms. In Section 4 we discuss a case study of one of the products and provide detailed insight into the challenges of productization in an IT services firm. Finally, in Section 5 we attempt to answer the main research question, by showing that the productization approach of Artz is indeed valid, but needed further detailing to be fully applicable to an IT services firm context. Perhaps most important is the finding that IT service firms need to redesign their strategies for harboring intellectual property, for capturing value in their software assets, and for productizing excellent customer solutions for a market.

2 Research Approach

In this research, we aim to evaluate and validate the productization process in an IT service firm. The advantage of taking the context of one IT service firm is that different projects in different phases can be studied. As the object of study is the project/product that is undergoing the productization process, we conducted theory-testing case studies within one context. The IT service firm is a large international service firm that, for reasons of confidentiality, is kept anonymous.

Seven projects were selected in the Netherlands. The projects were identified by senior management as being on the road of productization. These projects had typically been part of a development project for one customer, but with time similar requests came from customers and assets could be reused. The seven identified projects varied in age, number of customers, end-users, and team sizes. The main sources of data for the case studies were interviews and document study.

Semi-structured Interviewing: Semi-structured interviews are conducted as the second step of the research, with the aim of assessing the productization process of seven cases in the context of one IT service firm. In semi-structured interviewing, "a guide is used with questions and topics that must be covered. The interviewer has some discretion about the order in which the questions are asked, but the questions are standardized, and probes are provided to ensure that the researcher covers the correct material" [7]. The first step is the assessment of the maturity of the aforementioned Artz productization approach. This approach will be projected on seven case studies in one service company, with the aim of establishing the applicability of the productization process. The approach consists of following two steps:

1. ***Determine initial position:*** The productization assessment uses the SPM maturity matrix for determining the (present) SPM processes [18], the situational factors [2], and the applicable selected stages on the productization process for the to be analyzed products.
2. ***Gap analysis:*** A gap analysis identifies the distance from the initial position until being fully software product management oriented. Situational factors [2] are again used to determine the maturity levels for the product management processes.

Once maturity levels are known for each productized project, we determine which processes need to be implemented or improved. Based on the gap analysis, specific recommendations are identified to the IT service firm. In addition, we calculate the average percentage of each product management area based on the maturity matrix results obtained from all the business cases and compared the situational factors results of the cases with each other and finally, we reflected the productization profile of each business case to identify the weak and strong areas of the service business on their product management practices.

Literature Background. There are in the literature several scientific research studies available focusing on product software [21] such as product development [11, 8], management of software products [19, 5], requirements management [13, 4], release planning [14, 20], product line engineering [10, 12], product delivery [9], platform management [24], and so on. Despite the fact that there is an extensive research on the development of software as a product, some information concerning the development of product software is lacking [17]. Xu and Brinkkemper [21] state that by generalizing the experiences from product development, a body of knowledge needs to be established with theories, methods and tools. In this sense, from the product development perspective, this research takes the statement of Xu and Brinkkemper as a basis and aims to make a contribution to the literature by increasing the knowledge on the transformation process of software companies from developing customer-specific software to standard product software since this process is not widely studied. In addition, this research is a validation of productization process and it also improves and validates the defined set of dimensions by Artz et al. [1] for each stage in the productization process, which can be then used for the assessment of an organization's initial position based on the development of software as a product. Finally this research project creates many opportunities for further research on the transformation process from customer-specific development to standard product software development.

Research Validity. Yin [26] states that four types of validity are important in the exploratory case studies. First, ***Construct validity*** concerns the validity of the research method. Multiple data triangulation techniques may be applied in the sense that different stakeholders from different points of view will be interviewed to verify data. In the semi-structured interviews for maturity and productization process assessment, we made sure that the relevant concepts are correctly made operational (i.e., explained to the interviewee using a list of definitions) and measured. For the determination of maturity level of each product, where we asked standard questions, we applied for example the viewpoints of software developer, project manager and sales manager in order to get a better representative result. In addition, we also performed explorative interviews with the interviewees in advance with the aim to enlarge the reliability and to smoothen the proceeding of the interviews. The interviewees were read a standard text before the interviews as to make sure each of the interviewees was equally informed of the goals of the interview. Interviews were recorded and transcribed within 24 hours.

Internal validity is the extent to which a study's results are interpreted accurately. We tried to evaluate whether productization process was applicable in a service-oriented company. For internal validity of the research, we used a case study report template, inspired by the work of Jansen et al. [22]. Furthermore, we derived data from the cases using the SPM maturity matrix, thereby creating uniformity in the data.

Thirdly, ***External validity*** refers to how the data can be applied beyond the circumstances of the case to more general situations. We identified cases that differ from each other based on their situational factors with the aim to prove the applicability of the process. In addition, the selected interviewees differed a lot based on their experience and domain expertise.

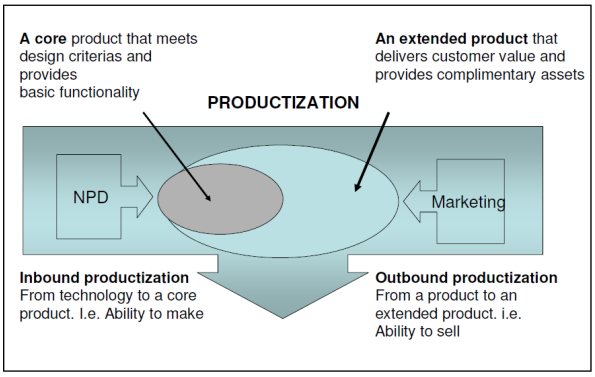

Fig. 2. Conceptual illustration of the productization process

Finally ***reliability*** indicates that the results of the study can be replicated. Since this research evaluates and validates the productization process, we performed the productization approach that is presented by Artz et al. [1] for the assessment product. Hereby all the procedures and steps have also been recorded. The business cases consisted of interview notes, documentation on maturity assessment and productization dimensions. We performed also maturity assessment for each product by making use of fixed standard questions from van de Weerd et al. [18]. The reliability is hereby guaranteed. When another interviewer performs the application of the model, the same results will be produced.

3 Productization

Productization means standardization of the elements in the offering. It includes several technological elements from the very early stages of designing a product (i.e. managing the requirements, selection of technological platforms, design of product architecture etc.) to the commercial elements of selling and distributing the product (i.e. delivery channels, positioning of the product/company and after-sales activities) [8]. According to Simula et al. [16], the term productization is mainly used in the context of software industries with the purpose to transform intangible services into more product-like, defined set of deliverables. A conceptual illustration of productization is shown as follows in Figure 2.

Artz et al. [1] identified six stages for the productization process, which describe the situation from customer specific development perspective to product software business perspective. The stages are described as follows:

- **Stage 1 - Independent projects:** This stage describes the situation of a service organization, which provides specific solutions per customer on project basis. According to Artz et al. [1], these projects are executed independently from each other and differ in budget, technology and functionality. They share barely any standard functions or features.
- **Stage 2 - Reuse across projects:** At this stage, reusability of existing components, functionalities and features is the main focus across projects. Artz et al. [1] state that reusing existing components from finished projects provides companies the advantage to increase the overall quality and reliability of software since they already have been tested within previous projects. At this stage, custom implemented features are still larger than standard features.
- **Stage 3 - Product recognition:** this stage describes the situation, where a company identifies the similarities of customers' wishes, which lead to the identification of a product scope. At this stage, the standardized part of the projects is larger than the customized parts because of the reused functionalities, components and features. This stage concerns also the decision moment to develop the identified product further on and to become market-driven business.
- **Stage 4 - Product basis:** Artz et al. [1] describe this phase: "A set of features that form a common structure, from which a stream of derivative products can be efficiently customized, developed and produced". A company starts at this stage to gather market requirements to determine the content of future releases. This means that a company needs to develop a long-term plan for the product.
- **Stage 5 - Standardized Product platform:** At this stage, the company changes toward market orientation and brings the emerging product to the market. Comparing to stage 4, the set of features, components and functionalities are increased through the product platform. The definition of this stage is as follows: "increasing the set of features that form a common structure and introduce releases, from which still a stream of derivative products can be efficiently customized, developed and produced".
- **Stage 6a - Customizable product:** This stage describes the situation, where companies offer their product software as customizable product for specific customers. The level of variability determines the applicability of the product in the market [23].
- **Stage 6b - Standard product:** This stage describes the situation, where companies offer their product software as a one-size-fits-all solution.

The Artz productization process is clearly defined, but too high level to apply to a detailed case, such as at an IT service firm. We therefore, in the next section, explore those dimensions that are specific to IT service firms that plan to productize.

3.1 Dimensions of Productization

Using literature, we have characterized the custom software service business and standard product software business. Since both types of businesses follow different approaches from a strategic point of view, we have identified the process areas that differ most. The process areas have been categorized in the societal, marketing and sales, company and development perspective (see fig. 2).

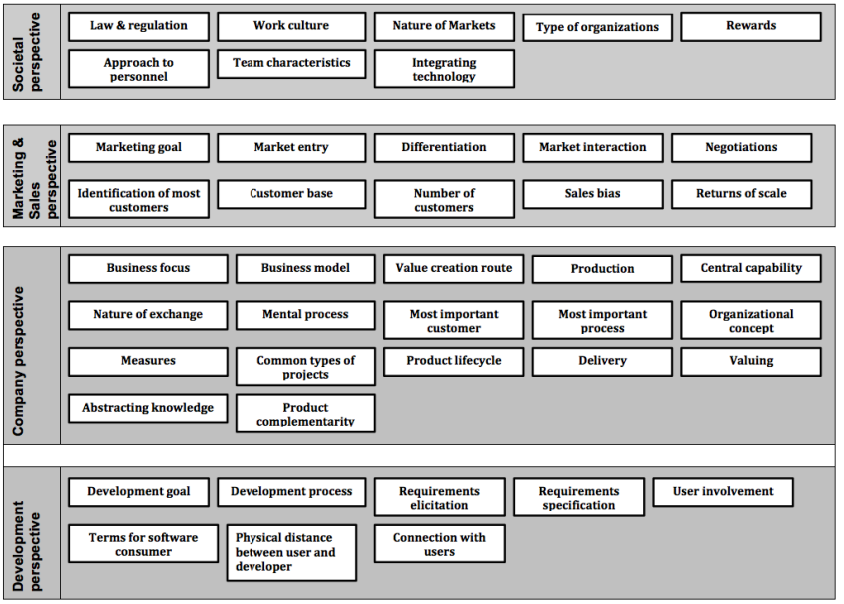

Fig. 3. Productization dimensions identified

Based on the dimensions, we improved the descriptions on the dimensions in the productization process and added three more dimensions (marked in gray) that may be relevant for the transformation process (see appendix A). The selected dimensions for the transformation are presented in table 1.

4 Evaluation Results

As mentioned earlier, the method that we have used for the evaluation is semi-structured interviewing, where we conducted seven case studies assisted by the interviewees from different domain and expertise in the service business. We tried to select the products from the different markets with the aim to take care of the internal validity of the productization process. Since the information related to the service company and the analyzed products had to remain confidential, we only provide the markets the products are intended for.

Table 1. Relevant dimensions for transformation

Dimensions	Customized software	Standard software
Software	Customized software project	Standard software product
Business focus	Meeting the customer needs within budget and time, contractual fulfillment	Gaining market share
Requirements gathering	Gathered from one customer	Gathered from whole market
Requirements selection	Select requirements per project (More or less fixed list of requirements)	Optical selected subset of requirements
Marketing goals	Interaction, relationship and networks	Product, price, place and promotion (4P's), branding and differentiation
Software development philosophy	Waterfall	SCRUM agile development
Lifecycle	One release, then maintenance	Several releases based on market requirements
Development teams	Project focused, people are assigned to multiple projects	Product-focused, self-managed, Involved in the entire development cycle
Stakeholder involvement	High external, barely internal	High internal, low external

Table 2. The assessed products in the context of the IT service firm

Potential products	Market sector
Product A	Telecom market
Product B	Telecom, Transport and Utility
Product C	Local government
Product D	Utility firms
Product E	Oil Companies
Product F	Local government
Product G	Local government

For reasons of brevity, we present one of the seven cases where we applied the productization approach. The other case studies can be found in a technical report [6].

4.1 Case Study

This business case is related to a customized product, which has been developed for the customers in the telecom market. The development has been done based on a European procurement and exists already twelve years on the market. Currently the

team focuses on developing this product as standard product from a software product management perspective. The aim is to become market-driven in the sense that the product can be released to different market segments.

Based on the productization approach, we followed the guidelines from Artz et al. [5] to identify the initial position and the best suitable position of the product in the process. The assessment has been performed together with the delivery manager, project manager and project director. The interviewees have been provided the productization process with the corresponding dimensions described in each stage and they were requested to fill in the applicable stages for the product.

Table 3. Demographics of interviewees

Inter-viewee	Years of experience	Current profile
1	23	Delivery manager projects: responsible for projects to deliver on time and within budget that meets client expectations.
2	17	Product manager; responsible for managing the products that fit the market needs, identify the needs and communicates to internal and external stakeholders.
3	12	Experienced project manager, service manager, and test manager in national and international environments, mainly in Telecom or Telecom related markets.
4	14,5	Program manager, project manager, responsible for devising, organizing and implementing projects that are complex in nature.
5	25	Delivery manager central government, infrastructure and environment.
6	12	Project manager with substantial experience in managing diverse ICT projects such as Data migration, infrastructural software development and COTS implementations.
7	20	Software Architect/Product manager
8	7	Software Architect
9	8	Sales & Solution manager for several clients within utility and Telecoms sector
10	5	Project manager at Technical Software Engineering

Based on the average score, we concluded that the initial position is at stage **one**. This has to do with the fact that there is still project-focus in the overall management of the product and the product is managed on a maintenance basis. The most important dimensions such as business focus, market goals and requirements gathering are not performed from a market approach but from specific customer needs. Another dimension 'stakeholder involvement' in the development indicates also this customer focus, where external stakeholder involvement is indicated 'high'.

For the gap analysis, we took into account the maturity assessment results and situational factors of the product to determine the best suitable position in the process by taking the determined initial position as basis. We also identified the current implemented and missing processes from the software product management perspective. The results indicate that the most of the problem areas of the product

concern portfolio management and product planning. Specifically, table 4 shows the missing capabilities that need to be implemented in the short term for the corresponding product.

We had already concluded that the initial position is at **stage 1** since the team follows a customer specific approach for managing the product and there are no processes implemented from a market-driven perspective. The most suitable stage for the product should be stage 6a since there will be always components that need to be customized to the specific customer's situation.

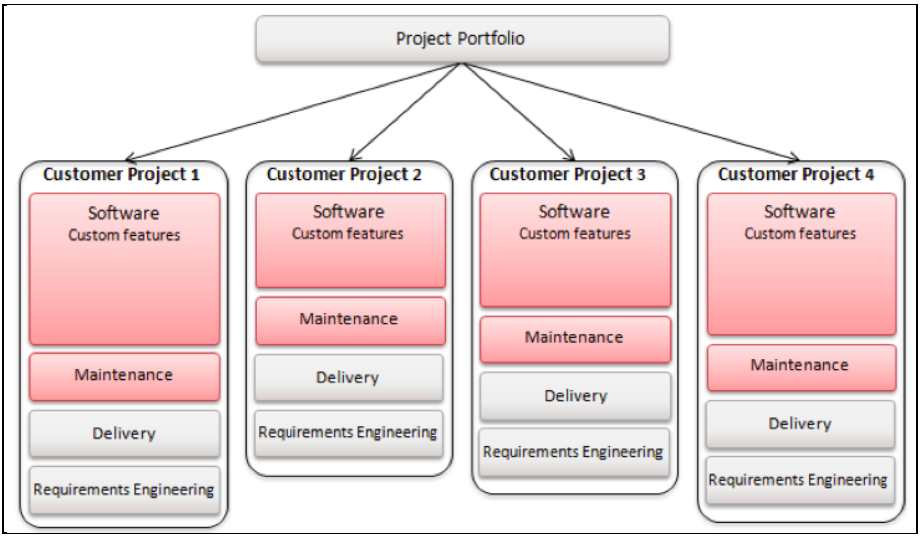

Fig. 4. Initial position

Table 4. Missing Capabilities Business case

Capability	**Business function**
Organize the requirements based on shared aspects	Requirements management
Automate of requirements since the requirements are already registered in a central database	Requirement management
Obtain formal approval for higher quality of releases (Company board involvement)	Release planning
Develop a short-term roadmap	Product planning
Identify themes to structure releases and roadmaps	Product planning
Analyze the product for different markets	Product planning
Follow the society and technological trends in coming years	Product planning
Identify the market trends	Portfolio management
Analyze the product lifecycle	Portfolio management
Innovate the product portfolio(s)	Portfolio management

4.2 SPM Maturity Assessments

In this section, we will present the results that we have made based on the overall SPM maturity results per analyzed product. The idea behind this calculation is to show how many capabilities have been implemented per product and what the overall percentage is of an implemented capability based on the analyzed projects/products. Based on the results, we categorized the products into *advanced products* and *emerging products*. Only two of the seven products were mature in their SPM processes and followed a market-oriented approach for the development. The other five were still immature to become fully market-oriented products despite the fact that they implemented some SPM processes quite well. An elaborated analysis on SPM calculations can be found in the technical report [6]. The calculated capabilities per SPM focus area are shown in table 5.

Table 5. SPM Maturity calculations

Focus area	Advanced (2)	Emerging (5)	Total (%)
Portfolio management	21,9	14,2	36,1
Product planning	19,3	15,3	34,6
Release planning	24,7	45,6	70,3
Requirements management	22	41,7	63,7

Most of the missing capabilities are in the area of product planning and portfolio management. The emerging products were quite mature in their internal development from the requirement and release management perspective but they do not consider the external factors from the roadmap and portfolio management since they simply miss the knowledge and experience in this area. The assessments gave us insight in the essential differences between the advanced and emerging products, which are shown in table 6.

Table 6. Differences based on SPM processes

Advanced	Emerging
Market orientation mindset	Service business mindset
Active market analysis	No or limited market analysis
Investment and support in the product	No awareness of product(s) in the organization
Considering ecosystems (partner involvement, make/buy decisions)	Limited partnering (based on components), high customer involvement in the product
Product lifecycle is present	No product lifecycle (contractual agreements)
Roadmap is present	Roadmap not possible (dependent on the investment from the company board)
Core assets are important and systematically identified and stored in a central location	Components are saved for reuse at the other customers
Requirements captured from all the internal, external stakeholders and the whole market	Only customer requirements count

4.3 Multi-dimensional Productization Profile

During the interviews, the interviewees have indicated the applicable stages for the corresponding products based on the relevant dimensions from the transformation perspective. In this section, we will present the (multi-dimensional) productization profile of one product based on the selected dimensions. We call this '*multi-dimensional*' since the applicable stages are selected on both extremes (for both service and product business) in the productization process at the same time. This can be confusing since the expectation is that the selected stages would be in one certain area.

For one specific product, we asked three interviewees (project manager, project director and delivery manager) to fill in the applicable stages, which are relevant in the productization process. The results are shown as follows:

Based on the selected stages, we can state that the product is still operating at the service business side but at the same time, there are two applicable stages selected from the product business side. One concerns the product being a (standard) customizable product and the other one concerns 'focusing on selecting requirements for the product (roadmap based) and subset of customer requirements'. All the other selected stages are in the area of the service business.

Table 7. Multi-dimensional productization profile

Service			**Hybrid**	**Product**		
Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6A	Stage 6B

To be able to underpin the underlying reasons for the selected stages on both service and product business side, we need to look at the specific situation of this. This product has been firstly developed based on the legislation (European procurement). It exists already 12 years and has been implemented at different telecom operators in a customized way. After having found unstructured areas regarding the management of the product for each specific customer, the team has brought major product revisions with the aim to bring more structure into it. The core platform has been hereby standardized for all the customers. In this sense, the interviewees have indicated that the product is standard with customizable layer. The customizable layer is the area that is integrated with the specific systems at the customers' side through certain interfaces. Standardization indicates that the team intends to make the product standard based on the market needs. In future terms, they are also conscious of the fact that the product can provide solutions to another type of markets however they are still at the initial stage of productization process.

When we look at the requirements selection, there are two stages selected for both service and product business side. One stage is related to selecting requirements from the customers (more or less fixed list) and another stage is about selecting requirements from all the customers (roadmap-based). This indicates that the team still follows the customer specific approach and manages the product based on the customer requirements but at the same time, they try to create the basis for market-driven development by selecting those requirements from the customers, which are relevant for the product. These requirements concern more the requirements from the customers and not the market(s).

5 Discussion and Further Research

During the interviews, there have been some discussion points on the productization process. One of the discussed points was related to the determination of the initial position of the products since some of the stakeholders pointed out that the stages have not been described in a S.M.A.R.T. (specific, measurable, achievable, realistic and time-bound) way. This made it difficult for experts to choose a certain applicable stage for the products. We would suggest performing more research on the description of the stages in the S.M.A.R.T. way by taking into account different service and product businesses to compare those companies with each other and gain insight in each stage of the productization process. Another point that is discussed is whether the required steps for each stage in the process included all the aspects that a service company needs to consider to become market-driven.

The productization approach that is developed by Artz et al. [1] is hard to apply in practice. This lies in the fact that the approach has been developed two years ago based on the relevant research subjects of that period of time (e.g. the gap analysis for determining the best suitable stage based on the research of Weerd et al. [17]). During this research, we could apply the approach based on the specific situation of the analyzed products by considering the situational factors, maturity assessments and the selected stages in the productization process, however more case studies can be performed to validate the approach by taking into account more additional steps that should be considered while productizing.

Finally, we identify a challenge in IT service firms: their traditional business model of selling projects and people *by the hour* makes it hard for them to invest in longer-running market-driven products. Typically, good ideas will be quelled before they get to a stage where they can be productized. We strongly recommend IT service firms to reconsider this strategy and to start creating intellectual property in the organization, for instance by collecting valuable marketable assets such as software products. Intellectual property increases the value of the company and selling licenses and subscriptions creates stability in a dynamic cash flow situation that IT service firms typically have. It is the responsibility of middle and upper management of such organizations to identify and invest in good ideas, marketable products, and online software services for a market.

6 Conclusion

This research concerned the validation of productization process at an IT service business. The validation has been performed twofold. The first validation was based on the assessment of products in the productization process, which had the potentials to be developed from the market-driven perspective. Regarding the assessment we took into account the situational factors and the maturity level of each product based on the product management areas: portfolio management, product planning, release planning and requirements management. We performed the second validation by considering the input of the interviewees on the productization process. Consequently, based on the results captured through the assessment and interviews held, we were able to answer our main research question.

We answered this question by assessing the productization process on seven specific products at an IT service business. From the productization perspective, we performed the steps such as analysis of situational factors, application of productization approach and analysis of productization profile of each analyzed product. In addition, we calculated the average percentage of the implemented capabilities per SPM focus area based on the maturity assessment results of all the products with the aim to gain insight in the strong and weak points of each product management area.

The overall results indicate that the (assessed) service business is not mature in developing products from a market-driven perspective. This is understandable since the core business is delivering services and meeting the customers' needs however the company is aware of the fact that based on their project portfolio, there are certain projects that have the potential to be developed as standard product software for certain markets. There are efforts made to gain insight in market-driven product development. Based on our analysis, we could see that the case organization scored high for the requirements and release management despite the fact that the focus hereby is still on the specific customer needs. Product planning and portfolio management areas are the weak areas that need to be improved. On the way to become market-driven, productization process is found applicable to use as a guide since the company aims to set up a structured product management process in place. It is essential to state that all the stages in the process have been recognized by most of the stakeholders however it is also indicated that the process steps through the stages still need to be developed on a S.M.A.R.T. way so that concrete measurements can be made to determine the initial and best suitable position of a product on the way to become market-driven.

Based on the current stage of productization at the case organization, we cannot expect the company to apply the process to its full extent since the basis needs to be set up for a product development approach. We recommend the company to put more focus on the stages three (product recognition) and four (product basis), where standardization is performed with a structured requirement and release management from a market-driven approach. In this sense, the overall customer focus needs to be decreased and the internal stakeholders need to be more involved in the product development.

Acknowledgement. We wish to thank the management of the IT service firm for supporting our research and the interviewees for their time and reflections.

References

1. Artz, P., van de Weerd, I., Brinkkemper, S., Fieggen, J.: Productization: transforming from developing customer-specific software to product software. In: Int'l Conf. on Software Business, pp. 90–102 (2010)
2. Bekkers, W., Weerd, I.V., Brinkkemper, S., Mahieu, A.: The Influence of Situational Factors in Software Product Management: An Empirical Study. In: Proceedings of the Int'l Workshop on Soft. Product Management, pp. 41–48 (2008)
3. Bekkers, W., van de Weerd, I., Spruit, M., Brinkkemper, S.: A Framework for Process Improvement in Software Product Management. In: Riel, A., O'Connor, R., Tichkiewitch, S., Messnarz, R. (eds.) EuroSPI 2010. CCIS, vol. 99, pp. 1–12. Springer, Heidelberg (2010)
4. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Nattoch Dag, J.: An industrial survey of requirements interdependencies in software product release planning. In: Proceedings. Fifth IEEE International Symposium, Requirements Engineering 2001, pp. 84–91 (2001)
5. Ebert, C.: Software Product Management. Crosstalk 22(1), 15–19 (2009)
6. Guvendiren, K., Brinkkemper, S., Jansen, S., Bleijinga, E.: Productization: Transforming from developing from customer specific software to standard product software, Utrecht University (2012)
7. Harrel, C., Bradley, A.: Data collection methods: Semi structured interviews and focus groups. National Defense Research Insitute. RAND Corporation, Santa Monica (2009)
8. Hietala, K., Kontio, J., Jokinen, J., Pyysiainen, J.: Challenges of software product companies: Results of a national survey in Finland. In: Proc. of the Int'l Symposium on Software Metrics, pp. 232–243 (2004)
9. Jansen, S., Ballintijn, G., Brinkkemper, S., van Nieuwland, A.: Integrated development and maintenance for the release, delivery, deployment, and customization of product software: A case study in mass-market ERP software. Journal of Software Maintenance and Evolution, 133–151 (2006)
10. Kang, K., Jaejoon, L., Donohoe, P.: Feature-oriented product line engineering. IEEE Software 19(4), 58–65 (2002)
11. MacCormack, A.: Product Development Practices that work: How Internet Companies Build Software. MIT Sloan Management Review 42, 75–84 (2001)
12. Pohl, K., Bockle, G., van der Linden, F.: Software Product Line Engineering Foundations, Principles, and Techniques. Springer, Heidelberg (2005)
13. Regnell, B., Host, M., Nattoch Dag, J., Beremark, P., Hjelm, T.: An industrial case study on Distributed Prioritization in Market-Driven Requirements Engineering for Packaged Software. Requirements Engineering 6, 51–62 (2001)
14. Ruhe, G., Saliu, M.: The art and science of software release planning. IEEE Soft. 22, 47–53 (2005)
15. Sawyer, S.: Packaged software: Implications of the differences from custom approaches to software development. European Journal of Information Systems 9(1), 47–58 (2000)
16. Simula, H., Lehtimäki, T., Salo, J.: Re-thinking the product – from innovative technology to productized offering. In: Proc. of the 19th Int'l Society for Professional Innovation Management Conf., June 15-18, pp. 1–13 (2008)

17. Van de Weerd, I.: Advancing in Software Product Management: An Incremental Method Engineering Approach. Utrecht University, Utrecht (2009)
18. Van de Weerd, I., Bekkers, W., Brinkkemper, S.: Developing a maturity matrix for Software Product Management. In: Tyrväinen, P., Jansen, S., Cusumano, M.A. (eds.) ICSOB 2010. LNBIP, vol. 51, pp. 76–89. Springer, Heidelberg (2010)
19. Van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: Towards a Reference Framework for Software Product Management. In: Int'l Requirements Eng. Conf (RE 2006), pp. 319–322 (2006)
20. Van den Akker, M., Van Diepen, G., Versendaal, J.: Flexible Release Planning Using Integer Linear Programming. In: Proceedings of the 11th Int'l Workshop on Req. Eng. for Soft. Quality, pp. 13–14 (2005)
21. Xu, L., Brinkkemper, S.: Concepts of Product Software. European Journal of Information Systems (EJIS) (16), 531–541 (2007)
22. Jansen, S., Brinkkemper, S.: Applied Multi-Case Research in a Mixed-Method Research Project: Customer Configuration Updating Improvement. In: Steel, A.C., Hakim, L.A. (eds.) nformation Systems Research Methods, Epistemology and Applications (2008)
23. Kabbedijk, J., Jansen, S.: Variability in multi-tenant environments: Architectural design patterns from industry. In: De Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., Van Mingroot, H. (eds.) ER Workshops 2011. LNCS, vol. 6999, pp. 151–160. Springer, Heidelberg (2011)
24. Jansen, S., Peeters, S., Brinkkemper, S.: Software Ecosystems: From Software Product Management to Software Platform Management. In: From Start-ups to SaaS Conglomerate: Life Cycles of Software Products Workshop (IW-LCSP 2013), p. 5 (2013)
25. Van Cann, R., Jansen, S., Brinkkemper, S.: Software Business Start-up Memories: Key Decisions in Success Stories. Palgrave Macmillan (2012)

Appendix A – Dimensions

	Service business			Hybrid Business	Product business		
Stages / Dimensions	**Independent projects**	**Reuse across projects**	**Product recognition**	**Product basis**	**Product platform**	**Customizable product**	**Standard product**
Software	Customized software	Customized software; consisting of large set of custom features across projects	Customized software; consisting of large set of standard features across projects	Standardized software with large customized layer per customer	Standardized software; emerging product; customized layer per customer	Standardized product software; customizable for customers by adding small customized layer	Standardized product, product is fully configurable
Business focus	Meet the customer needs within budget and time, contractual fulfilment (customer satisfaction)	Meet the customer needs; amortize costs and reduce time to develop and increase quality/efficiency by software component reuse	Meet the customer needs but create product(s) based on standard features and focus on market potential for the product(s)	Meet the customer needs but create a generic product platform for a market; focus on partnering and long term planning to gain market share	Meet the customer needs but focus on bringing the product(s) for large set of customers in the market, identify product lines, determine your pricing, positioning and product strategy	Bring the product(s) to the market as customizable for each customer and provide support services.	Bring the product(s) to the market and increase market share by selling licences
Market goals	(Not important) it is based on Interaction, relationship and networks, no reserved budget for marketing	(Not important) obtain components from different vendors to integrate with for customer specific solutions	Marketing becomes important. Identify target market, define product goals and determine the entry mode and make win/loss analysis	Discover problems in the target market by interviewing customers, recent evaluators and potential customers to capture market requirements for the product	Identify competitive and alternative offerings in the market and develop strategy for winning against the competition (assess the technology inside and outside the organization)	Focus on branding and differentiation and aim at selling services based on business-to-business relationship	Focus on product, price, place and promotion (4P's) and 4C's (Customer, Cost, Convenience and Communication) and branding/ differentiation
Lifecycle	Maintenance per customer	Maintenance per customer based on structured component-based software engineering process	Maintenance per customer	Maintenance per customer (including product platform)	Releases designed per customer	Releases based on fixed release dates; equal for all customers (not customizable)	Releases based on fixed release dates for the entire market
Software development philosophy	Waterfall	Waterfall, Component-based	Waterfall, Component-based	Waterfall combined with iterative development	Waterfall combined with iterative development	SCRUM (Agile) development	SCRUM (Agile) development
Requirements gathering	Gather requirements for each customer	Gather requirements for each customer and match the requirements with reusable components	Gather requirements from all customers and store them in one central place	Gather requirements from all customers and start involving internal stakeholders	Gather requirements from all the internal and external stakeholders; start looking at market trends	Gather requirements from entire market; all internal stakeholders and all customers	Gather requirements from entire market; all internal and all external stakeholders
Requirements selection	Select all customer requirements per project (more or less fixed list)	Select all customer requirements per project (more or less fixed list) and select the custom reusable components/features to implement the requirements	Select all customer requirements per project (more or less fixed list) and identify the standard existing features/components that meet the requirements	Select requirements for product platform and select (large quantity of) customer requirements	Focus on selecting requirements for product (roadmap-based) and subset of customer requirements	Optimal selected subset of market requirements (roadmap based) and requirements for customization	Optimal selected subset of market requirements (roadmap based)
Development teams	Project focused, people are assigned to multiple projects	Project focused, people are assigned to multiple projects, developing in a way to enable reusability	Project focused, people are assigned to multiple projects, start to form product team	Product focused, developing the core product platform from technology	Product focused, sharing product vision to bring the product to market	Product focused, sharing product vision to sell the product on market	Product focused, sharing product vision to sell the product to market
Stakeholder involvement on development	High external, barely any internal	High external, barely any internal	High external, low internal	High external, medium internal	Medium external, medium internal	Medium external, high internal	Low external, high internal

Software Development as a Decision-Oriented Process

Jarkko Hyysalo[1], Markus Kelanti[1], Jari Lehto[2], Pasi Kuvaja[1], and Markku Oivo[1]

[1] University of Oulu, Department of Information Processing Science, M-Group,
P.O. Box 3000 FIN-90014 Oulu, Finland
{Jarkko.Hyysalo,Markus.Kelanti,Pasi.Kuvaja,Markku.Oivo}@oulu.fi
[2] Nokia Solutions and Networks Oy, P.O. Box 1, FI-02022 NSN
Jari.Lehto@nsn.com

Abstract. Developing software systems is a challenging business with short development cycles, changing needs, and unstable processes. Processes must deliver products that meet the customer needs and provide value for the stakeholders. There is no one way of achieving the development goals; instead, alternative routes should be possible within the boundaries of acceptable performance. Software development is therefore a set of problem-solving and decision-making activities. The problem is how to support the decision-oriented process, and how to provide justification, rationale, and how to provide the information that decision makers need. Case studies in the automation and telecom industries revealed that understanding the development process as a decision-oriented process, and controlling and coordinating the work through decision points offer an approach that addresses several challenges. The findings of this study offer new insights for scholars and practitioners.

Keywords: Decision-making, information flow, software development process.

1 Introduction

Software systems are becoming more complex, and consequently, developing software systems is becoming increasingly challenging. Product development is a complicated process with tight time-to-market requirements, and changes in markets and technologies require flexibility in manufacturing strategies [1]. Developers face several challenges, such as making complex decisions, solving problems, handling vast amounts of information, creating shared understanding, and sharing information and knowledge [2]. Products are also typically developed in collaboration between different stakeholders [3], [4], [5], as different skills are required. Each stakeholder contributes to the common goal, so the stakeholders must understand what is expected from them. Interchange of data between customers, suppliers, and authorities is needed [1]. Properly defined goals and criteria help create a shared understanding of the work to be done and coordinate the goals and work efforts. Then, developers need to solve the "problem" of how to reach the goals. Clear goals and accurate decision criteria also help match the work to the needs.

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 132–147, 2014.

The software development process could be modeled as a set of problem-solving activities [6]. For example, Aurum and Wohlin [7] emphasized that problem solving is, in essence, decision-making. Thus, software development is a decision-oriented process [8]. Decision-oriented process modeling has been suggested as a step toward human-centered process management [9]. Addressing the decision-oriented nature of software development, and providing the necessary support for collaborative development require recognizing the importance of decision-making, synchronization, and coordination. These factors should be integrated into development processes and practices. In this article, we study how to approach decision-oriented development: how the decision-oriented aspects and the process can be organized, and what kind of elements it should contain. We formulate the research problem as follows: How does decision-making control and guide the software development process?

We address this question through case studies that examine the issues and solutions that push toward decision-oriented work. The cases form the basis for solving the research problem. In our study, we found that decision points (DPs) are one way of addressing the research problem, as they not only define what should be achieved, but also express why it is needed, providing the rationale (argument) and reason (purpose) for the work. DPs also coordinate and synchronize the work.

Two cases within two large companies in the automation and telecom industries were set up. Forty-six experts and managers were interviewed, covering the development process from several perspectives and phases. We identified common trends in the processes. The results showed that decision-making plays an important role. We found a method for addressing the decision-oriented nature of software development and structuring the problem-solving activities. We suggest that a decision-oriented process provides the decision criteria and degrees of freedom required for innovative problem solving.

This paper presents a method for providing the support needed for development activities. We do not focus on decision-making activities, models, and theories per se, but instead show how decision-making is an integral part of the development process and how decision-making and decision criteria justify, guide, and control developers' daily tasks. The remainder of the paper is structured as follows: In Section 2, we examine related work; in Section 3, we present the research process; in Section 4, we present the empirical study with its findings; in Section 5, we discuss the elements of decision-oriented development process; and in Section 6, we conclude the study and discuss the implications.

2 Related Work

Processes are meant to ensure that activities in an organization are performed consistently and reliably [10]. The four primary components are objectives, tasks, performers, and constraints [11]. However, it is not easy to plan the work in detail beforehand, as development in the turbulent world today is characterized by constant changes in goals, work, resources, environment, etc. Trivial cases and simple problems can be solved with the top-down approach, while non-trivial problems are characterized by deviations from the predefined top-down approach [12], [13].

Design problems often result from ill-defined goals and evaluation criteria, which may cause new or changing requirements and deviations from goals and predefined process schemas [12]. Furthermore, the exact order in which activities are executed is not necessarily important, as interaction with the environment, the activities, and underlying business logic sets the order, instead of predefined, static process schema [14]. As long as the appropriate design is discovered, the process should be opportunistic; then, the design process can be top-down [12], [13].

Since processes are complicated and not all circumstances can be predicted, uncertainty and changes prevail in software development [12], [15]. Changes and unexpected events require creativity and human problem-solving skills to overcome and solve them [16]. The use of opportunistic design behaviors has been proposed [12], where processes are modeled at a high level, and knowledge-intensive tasks are represented as black boxes [17]. In sum, no one method should lock developers into a strict order of activities that hinders opportunistic insights; there should still be adequate support for developers' activities [12].

In addition, collaborative development necessitates a shared understanding of what needs to be done to reach the desired objectives. Reaching the desired objective is a problem-solving activity that requires decision-making. Iivari et al. [18] discussed the three decision elements in which 1) a goal or target provides feed-forward information that indicates the desired state of the system or a subsystem and performance expectations, 2) state information indicates the current state of the system or subsystem, and 3) information is the result of data interpretation, where data as a decision element are related to a specific context.

However, Zhuge [19] proposed that problem-solving processes should be managed at two levels: 1) the work level, which includes planning tasks, scheduling resources, and managing the workflow management, and 2) the cognitive level, which includes learning problem-solving knowledge, skills, rules, methodologies, etc. In practice, decision-making is complex, owing to the limited capacity to understand everything, the limited time in which to make decisions, the limitations of the schemas, and the multitude of attributes to compare (e.g., strategy choices) [20], [21], [22]. Thus, decision-making is an integral part of the development work; problem solving must be structured and decision-making supported.

Much of the problem-solving effort is actually in structuring the problem [23]. To summarize, humans intuitively understand a good design emerging from a well-defined problem [24]. A problem is well defined if "the initial state, the goal, and a set of possible operations are available to reach the goal from the initial state" [24]. In addition, the problem type should be identified, as it is then easier to find a solution [25]. The problem solver must understand the reasoning behind the problem, and the concepts and relationships of the problem elements [26].

Software development is a decision-oriented process, and there are several ways to approach decision-oriented development, such as decision-making processes [27], optimization [28], selection in product design [29], negotiation networks [30], and collaborative design decision-making [31]. Many articles are very theoretical discussing mathematical models on decision-making and decision-making theories, and the findings are hard to apply in practice in the software development process.

Software development decisions can also be approached from other directions, such as knowledge management [8] or management decision gates [32]. However, regardless of the development approach, the decision-making and decision criteria must be integrated into the development process [2]. Integrating them into the development process with support for decision-making would address the decision orientation, guide the developers' daily activities, and create a structure for the process and problem solving. This way, the decisions and decision criteria justify the work and define the goals for developers, while the dependencies between the goals define the dependencies of the activities. Understanding the development process in this way is what we consider is missing in the literature.

3 Research Process

The research consisted of studying the software development processes of two case companies acting in different domains. Company A is one of the largest information and communication technology providers in the world in its field. Company B is an organization that develops process automation systems for industrial users. The companies are distributed across the globe and collaborate with numerous different stakeholders and organizations. The case companies use traditional and agile development methods, depending on the product. In the companies' documented processes, there are defined checkpoints where certain maturity criteria must be met in order to let a particular working item continue to the next phase; however, these checkpoints are loosely followed, depending on the project manager. Activities are further split into tasks, which are planned according to the intended output. After a task has been implemented, an activity continues with the next task until all of the work has been conducted for a certain checkpoint. After the results are accepted, the work continues.

The research process follows the guidelines presented by Runeson and Höst [33], as shown in Fig. 1.

Fig. 1. Research process [33].

In the design phase, the cases were defined, the objectives and research problem and the theoretical background were determined, and the research methods and sources were chosen. The preparation phase included developing the questionnaire, deciding whom to interview, and agreeing on the procedures. In addition, the questionnaire template was piloted and reviewed by each company representative. During the collection phase, data were gathered from qualitative interviews, archive material, and the literature. Each interview was transcribed, summarized, and sent to the interviewee to check and correct. Nvivo 10 was used to store the interview data and to help facilitate the analysis process. The interview data was first coded based on the interview questions, and then analyzed and coded according to themes emerging from the interviews. Each company was first examined independently, and the data

was then cross analyzed over the different cases and domains, and compared to the existing literature. Workshops were arranged at the companies to discuss the findings and results. Feedback from the workshops was incorporated in the final analysis.

The study consisted of 35 interviews at Company A and 11 at Company B. The companies participated in the AMALTHEA project, the focus of which was to build an open-source tool platform that forms a tool chain to help develop multi-core embedded systems in automotive, industrial automation, and telecommunication domains. The cases were chosen as they could all provide potential data related to the research question, as proposed by Yin [34]. Experts and managers from different levels of the organizations from multiple sites and countries were interviewed. We chose a semi-structured thematic interview approach. Different questionnaires were targeted toward each company and different development phases, with relevant questions concerning the interviewees' work. The interviews were qualitative, allowing the interviewees to explain and clarify their views as freely as possible. As a result, the interviews represent the development processes in the case companies.

4 Findings and Analysis

This section presents the findings and analysis that promote the decision-oriented approach and address the research problem. The findings are organized under three themes that emerged from the interview analyses: information flow, development process, and decision-making. The findings promote the importance of accurate information, decision-making, synchronization, coordination, and understanding of the needs and objectives of the work. In the following three sub-sections, we discuss the findings derived from the case companies, and we analyze and compare them to the literature.

Table 1 summarizes the findings and analyses. Some of the items may seem self-evident; nevertheless, as the companies are struggling with those issues, we feel the need to mention these points. Our analysis suggests that a decision-oriented approach is needed.

Table 1. Drivers for decision-oriented software development process.

Topic	Drivers for decision-oriented process	Related work
Information flow	Provide clear goals and objectives	[35], [36], [37]
	Provide up-to-date information about process and tasks	[6], [38]
	Provide sufficient information for decision-making	
Development process	Provide an awareness of the development process	[38], [39], [40]
	Recognize the context of activities and tasks	[41], [42]
	The development process must be flexible	[12], [13], [14]
	The process must support and guide the work	
	Regular checkpoints to obtain feedback and status, and to synchronize the work	[3], [43]
Decision-making	Provide decision-making awareness	[7], [44]
	Provide acceptance criteria	
	The process should have regular reviews	[45]
	Clearly define DPs to control the work	

4.1 Information Flow

Based on the interview results, the goals and objectives should be communicated early and during a single task or assignment during a development process. Customers (both internal and external) expect the results of the process or sub-processes to match the agreed terms. In addition, the rationale for why a certain piece of work should be done or information should be provided helps to justify the resources spent for doing a certain activity and to focus on and prioritize resources. Therefore, **clear goals and objectives should be provided** since they help the collaborative work and improve efficiency by reducing redundant work [35]. The organization's strategies, goals, and motives should be included in the work [36]. In addition, sufficient knowledge and understanding of others' problems aids effective communication between the stakeholders [37].

Interviewees mentioned that there is no overall view of the work status. Managers, developers, and customers need **up-to-date information on the process and tasks** i.e., the progress, estimates of delivery dates, resources, costs, etc. This is not only a tool-related issue; the process must also help gather the data easily. However, current processes do not provide all of the necessary data. For example, in Company B, data are provided at the end of each development phase. However, since the phases may be very long, the status during those phases is not available. The implementation phase, in particular, may be long.

The literature also advises that the state of the project activities and tasks must be available and understood to react to changes and to build a shared understanding [38]. The state of the tasks can be followed through the decision-making process. When a problem is solved, the task moves forward, while unsolved problems remain on the agenda [6].

Interviewees had a clear need for appropriate information on which to base their decisions. Currently, the information was insufficient or stored in a way that was not accessible or useful. For example, requirements were not sufficiently detailed for decision-making, and rationale, processing, and acceptance criteria were not adequately defined, which made decision-making problematic. Interviewees proposed that **sufficient information for decision-making must be provided.**

4.2 Development Process

The interviewees suggested that problems emerge in the case companies because of a lack of understanding about what others' activities required and a lack of feedback. Different actors have a different perspective on the product accompanied by varying understandings of a product's lifecycle. In addition, a generalized view of projects and project portfolios seemed difficult. Even if the processes are defined and documented, the general feeling among the interviewees is that the process does not match the official guidelines, owing to different activities or practices at the team, product, or even organizational level.

An awareness of the development process and activities is important since processes and phases must be visible to provide information about the context and

dependencies between tasks and work items. Understanding the relationships helps people understand the entire development process. Being aware of the context of the work and others' actions helps developers structure their own interactions and cooperative processes, and provides a context for one's own activities [38], [39], [40].

In some cases, the interviewees saw a lack of cooperation between different development phases and processes. In a common case, most of the effort goes to producing immediate results at the expense of the final outcome—the sellable product. Interviewees suggested that understanding the entire process and enhancing cooperation would improve the work results.

Interviewees stressed that it is essential to **recognize the context of the activities and tasks**. One must know all about a requirement, including its context and environment, and the motives and needs behind the requirement to wholly understand what is needed to address that particular need. Interviewees mentioned that it is hard to see the motives for things that do not directly provide profit for the company, such as savings or improvement of the company's image. In addition, the literature sees context specification as an important part of defining goals and preventing drawing wrong conclusions [41]. In addition, Molina and Olsina [42] suggested that context information is important for measuring and evaluating activities.

Not all development projects are similar, and adaptability is required. According to interviewees, **the development process must be flexible**: "It is important that the processes enable all the stakeholders a flexible way to do different things in different ways (Interviewee, Company B)." In addition, defining a process too strictly hinders innovation. The development process requires a lot of creative work. For example, one of the interviewees said that developers have to do a lot of thinking, and they must be able to model large entities in their minds. In addition, there must be room for adjustments to account for unexpected situations. These findings are supported by the literature [12], [13], [14].

However, since the interviewees hoped for flexibility, guidance must direct the work toward a focused, common goal. Interviewees suggested that **the process must support and guide the work** and include checks to ensure that the overall goals are being implemented. If the guidance is strong enough, the developers can work toward their goals quite freely and perform activities simultaneously. Interviewees also expressed that receiving constant feedback on one's activities is necessary. After further analysis of the interviewees' main problems within the process, it became clear that guidance helps developers to provide information in the correct context. Whether involved in decision-making, analysis, or practical work to create a product, the common theme was that developers had to confirm or ask for further clarification about the information they received. In most cases, the information was out of context, written at too technical or abstract a level, or lacked the necessary information for what the task developer was doing. Providing guidance on what information is needed in the tasks would therefore be useful.

There must also be **regular checkpoints to synchronize the work, identify the status of the work, and obtain feedback**. When the work is divided into loosely coupled activities and tasks, and the order of operations is not strictly defined, the activities must be coordinated to produce a coherent piece of work. Checkpoints

ensure whether the development work is on the correct path. A suitable interval for checkpoints should be found to provide a balance between control and freedom. Interviewees liked the idea adopted from Scrum to have checkpoints every three weeks. Checkpoints are also proposed in the literature, e.g., in the stage-gate process [3] or for processing product data [43].

4.3 Decision-Making

In general, interviewees knew there was an established general framework for the entire development process divided by milestones or checkpoints. These milestones or checkpoints served as DPs where information was collected and decisions were made to determine whether a customer request should be further analyzed or whether a business plan for a certain feature is feasible and should be made. However, the actual criteria for the decision or the decision-making process in DPs were seen more as an ad hoc review meeting in which the items are discussed and decided based on discussions, and no documented processes are used. Some interviewees expressed the need for more rigorous decision-making with defined decision criteria. The main rationale behind this request was the need to communicate the results to other stakeholders. However, interviewees pointed out the need for awareness in these decision-making points.

The literature also promotes **decision-making awareness** and clearly defined **acceptance criteria**, stating that the decision-making process and forums should be transparent, which would raise developers' awareness of the persons working on a particular decision. Keeping track of decisions, rationale, and decisions' effects on software products is also advised [7], [44].

Review practices in the case companies were included in a normal process; only the format and frequency differed depending on the development domain, method, and developer's experience. Interviewees in the case companies use these reviews to obtain others' opinions and evaluate whether they missed something or something has changed in the product, technology, business perspectives, or company's strategy. In addition, the review meetings were also used to inspect existing work and make a decision (to continue, do additional analysis, terminate, etc.). In general, there were at least two types of DPs: process-level decisions on features and products and development-level decisions on aspects in features and products. Interviewees were quite familiar with Scrum and appreciated regular review meetings in short intervals at the team level. Managers appreciated process-level DPs, while development-level DPs were needed to stay up-to-date with development effort.

Based on the findings, the **process should include regular reviews**. Regular reviews guarantee that the results and progress are checked, and stakeholders understand the current situation. DPs are natural places to hold the review process. Review processes are nothing new in software development, and various methods are available [45], further emphasizing this finding.

Clearly defined DPs to control the work as well as defining the interdependencies of DPs were requested. DPs are natural points for checking progress and results, and to integrate the work. For each DP, there should be a defined

input and output, accompanied by information about tasks, resources, objectives, decision criteria, and the context. In this way, the process follows a goal-oriented approach, the measurement and evaluation are done in DPs, and activities within DPs focus on satisfying the specified needs. The results of each activity and task are presented for decision makers as proposals to be evaluated against the decision criteria. Criteria and objectives are presented through the DPs, and the work is divided into suitable pieces of work.

5 Toward Decision-Oriented Work

In this section, we discuss the decision-oriented nature of software development, based on the findings and analyses of the previous section. By using DPs and decision criteria, the development process can be better controlled, synchronized, and justified.

In the decision-oriented approach, the process is described using process elements that are the highest-level concept in this framework. They are designed to have clearly defined input(s) and output(s). Input is any information the process element receives, which works as a starting point for the activities inside a process element. Output is the defined information content provided by the process element. By default, the process element does not tell how information is processed or how work is done inside the element itself. The purpose of the process element is to describe an upper-level activity that can produce a meaningful piece of information. Each process element has a purpose, and it is important to provide a valid argument and reasoning why the work is needed for that particular element.

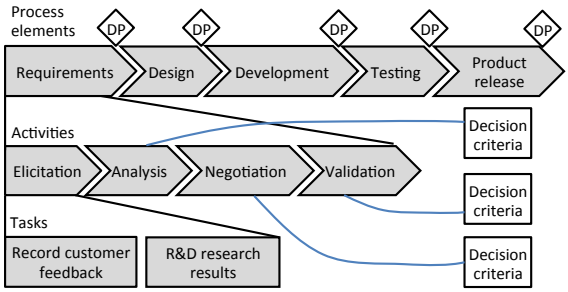

Fig. 2. An example of process showing the context of the process elements, activities, and tasks

Fig. 2 presents an example of a process. Process elements are divided into activities and then further divided into tasks that progress toward completing the work. Each element, activity, and task belongs to a certain context. In addition, the entire process is in a context influenced by the organization, environment, stakeholders, etc. The following elements form the decision-oriented approach:

Process elements have a purpose that fulfills the goals and objectives of the processes. The activities and their order within process elements can be selected and changed according to needs at hand; thus, the process approach can be top-down or bottom-up. Work done and resulting work items are available for others to continue

the work. The network of dependencies tells what results and work items are needed as an input for the task at hand.

DPs are used to coordinate and synchronize the work. A DP defines the information content the process element produces as an output for other process element(s). The DP includes a list of decision criteria that must be fulfilled before the information content produced in the process element can be sent to other process elements. The DP justifies the work to be done and describes the actual information content and the information needed to make a decision.

Activity produces the information needed to meet the decision criteria. Activity is an abstract entity that contains only the tasks necessary to provide the information. An activity is composed of tasks (pieces of information that are needed in a DP according to the decision criteria) and dependencies.

Task is an operation that cannot be divided into smaller pieces of work. A task belongs to an activity and describes a concrete task that must be done. A task is composed of the information field in decision criterion it aims to provide, as well as the information about the tool used to complete this task.

Process elements are related to the entire process, and activities and tasks relate to the process elements. Each process, element, activity, and task has a goal and a purpose. During each phase, work is done to fulfill the decision criteria set for each DP. Justification for the work is described in the DPs, and the decision criteria define when the purpose is achieved. Furthermore, DPs are natural places for synchronizing sequential and often concurrent activities.

The process, methods, and practices as well as the workflow used to implement and enact them should support freedom in the order of activities and in the implementation of practices and strategies. This creates a flexible and adaptable workflow and enables developers to implement their activities and tasks as they see fit. The development activities are guided by objectives, work requirements, constraints, and resources, which are in practice addressed through DPs. It is not necessary to define complete and detailed relations of tasks consisting of each alternative order of tasks; only the mandatory input/output sequential orders are defined. The activities are drawn as black boxes as much as possible to allow the developers enough freedom to choose their work style, leaving room for innovation and intuition, and for supporting opportunistic design processes.

The aim is to have a simple, generic framework that is situation-independent; see Fig. 3. Situations are reacted to via decision criteria. Acceptance criteria are transformed into decision criteria, which guide and drive the task planning along with goals. The decision criteria of each DP are converted into guidelines for the designers and reviewers. Decision proposals are prepared after activities consisting of tasks are completed, and decision proposals are sent for approval. The approval decision is the final activity in a DP, determining the output of an element. When an item is approved, it is ready for the next phase. At the approval, the results are compared against the process and product criteria. Each decision criterion should be fulfilled before the output is sent to other process elements. In addition to product-related criteria, process-related criteria provide principles, guidelines, and directives that set criteria or constraints for implementation. Process criteria execute more long-term effect controls and controls that are in effect over the product lines, possibly through the entire organization.

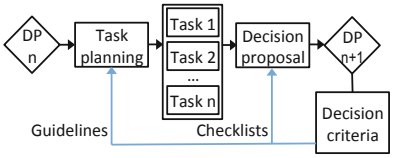

Fig. 3. Decision criteria guiding the work

Decision criteria belong to a DP describing information that must be known in a DP before a decision can be made. The information content is created in an activity. The intention is that all decisions are made according to the defined criteria of a DP. Relevant stakeholders, who are also responsible for keeping the criteria up to date, set the decision criteria. The stakeholders can change, update, and remove criteria but they cannot remove parts of the process, for example. The work results will always address the needs when the criteria guiding the development are up to date.

It may also become necessary to check if the activities and tasks are relevant for the entity under analysis. If it is noticed that something does not belong to a certain context, it must be considered whether the activity or task should be removed completely or whether it would be more suitable in another element or context. The need for new activities and tasks should also be considered; thus, a review process should be set up.

Fig. 4 shows an example of a process diagram. Directional arrows link the process elements and define the information flows between the process elements. A process element can have multiple incoming or outgoing arrows. Arrows define the direction of the information flows between process elements, and each arrow that leaves from a process element has to have a DP where it originates. Activity describes work that is necessary for creating the required information for a certain decision criterion.

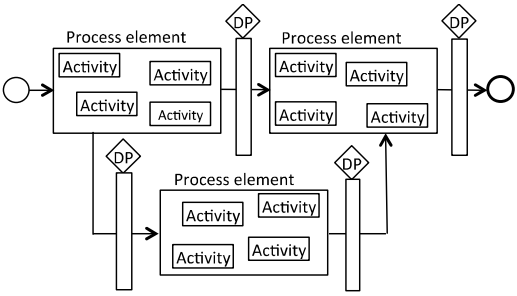

Fig. 4. Process elements and their relations

Fig. 5 illustrates how activities, DPs, and tasks are related in a process element. The left arrow describes the input that comes in to a process element. This process element has three activities, and their decision criteria are defined in a related DP. All information for activities A, B, and C must be produced to make a decision and submit the results to other process elements. The purpose of the links between activities and DPs is to show the relation between these items. For example, activity A in Fig. 5 depends on incoming information as well as information from activities B

and C. The figure does not state which activity comes first; it only describes how the activities depend on each other and where the information comes from to form the set of activities. In this example, the information for the three tasks for activity C is defined in the decision criteria for that activity.

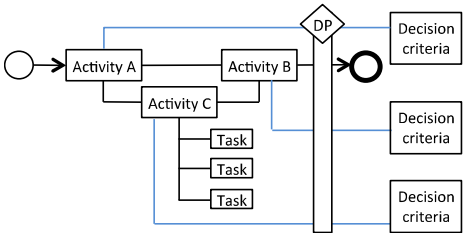

Fig. 5. An example of activities and tasks within a process element

6 Conclusions

This paper shows how software development can be understood as a decision-oriented process. We presented the drivers for a decision-oriented process, organized under three themes of information flow, development process, and decision-making. The findings and analyses address the research problem, and provide the basis for further discussion on the decision-oriented nature of software development. The literature has pointed out that this area of research is important and has not yet been properly addressed; to our knowledge, no decision-oriented process is available in the literature in the way we present it. Our case studies show the benefits of understanding the development process in a decision-oriented way. The decision-oriented approach informs users how a development process works at different abstraction levels and describes how the information flows, what kind of information is needed and why, and how the information is processed at different levels.

The main idea is that DPs and decision criteria define why the work is done, and guide and control the work. No specific order for the activities is enforced, and activities and tasks can be changed according to the requirements. Although the goals and purposes define what needs to be done, clearly defined acceptance criteria are needed from the process and product points of view to guide the decision-making activities. Product-related acceptance criteria are derived from the requirements describing the minimum effort needed to implement a requested artifact. Alternatively, process-related criteria are a set of predefined rules defining the fulfillment criteria for tasks from the development process point of view, e.g., relating to input required for subsequent tasks. Defining the process in this way provides the following benefits:

- The process is described at the general level, where the process elements are linked as a directed network. By using DPs to determine all outputs and the content of each element, the process is guided by the DPs, and the output is always defined by the decision criteria.

- As the process element output is defined by decision criteria, the criteria can be utilized to generate reports from any process point at any given point of time. I.e. comparing the current status of tasks to the decision criteria. In addition, as information flows through the process, a report can be produced for any single data item, dependency relation, history data, and so on.
- A progress report for a single item in the process can be easily generated because of the dependency on the DPs decision criteria that informs the current state of progress to reach that DP.
- Since DPs determine what activities and tasks are necessary, any unnecessary work can be eliminated from the process, as it is not required to satisfy the information need in a DP. If additional work is needed, the model forces the users to re-evaluate the process and add or remove parts.
- The process is easily interpreted, and it enables a quick understanding of how the process works, how information flows, and the purpose of each process element, activity, and task.
- It also determines what kind of tool support is needed for process elements, activities, and single tasks, as requirements are set at the individual developer level and at the process level by describing the necessary work that must be done and how the data are used and transferred inside the process.

The results should interest academics and practitioners; they indicate that the decision-oriented approach is useful for supporting collaborative software development. The study provides valuable insights for academics, as it seems that the decision-oriented development process is not used in industry in a similar way and context as in this study; reports of such attempts could not be found in the literature. The study also lays the groundwork for further scholarly inquiry, including validating the findings in phases other than requirements development and other domains. It is also our intention to try the approach in other domains.

For practitioners, the decision-oriented approach provides a better understanding of the context of the work by defining the real needs of stakeholders, processes, activities, and tasks. It will lead to better product quality and shorter development times, primarily because the results of work activities and tasks will fulfill their purpose better and provide less waste. Taking a decision-oriented approach to software development, developers are able to address the dynamic development environment of today's software business, and tackle the inevitable changes. Different abstraction levels are all acknowledged, and goals and high-level objectives are presented, with justification and rationale for the decision-making.

Acknowledgments. This research is supported by the European ITEA2 program with national funding from Tekes (the Finnish Funding Agency for Technology and Innovation). The authors would like also to thank AMALTHEA project partners.

References

1. Helo, P.: Managing Agility and Productivity in the Electronics Industry. Ind. Manage. Data Syst. 104, 567–577 (2004)
2. Hyysalo, J., Lehto, J., Aaramaa, S., Kelanti, M.: Supporting Cognitive Work in Software Development Workflows. In: Heidrich, J., Oivo, M., Jedlitschka, A., Baldassarre, M.T. (eds.) PROFES 2013. LNCS, vol. 7983, pp. 20–34. Springer, Heidelberg (2013)
3. Cooper, R.G., Edgett, S.J., Kleinschmidt, E.J.: Benchmarking Best NPD Practices-III. Res. -Tech. Manage. 47, 43–55 (2004)
4. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.H.: Engineering Design: A Systematic Approach, 3rd edn. Springer, London (2007)
5. Zeidler, C., Kittl, C., Petrovic, O.: An Integrated Product Development Process for Mobile Software. Int. J. Mob. Commun. 6, 345–356 (2008)
6. Wild, C., Maly, K., Zhang, C., Roberts, C.C., Rosca, D., Taylor, T.: Software Engineering Life Cycle Support - Decision-Based Systems Development. In: IEEE Region 10's Ninth Annual International Conference TENCON 1994, pp. 781–784. IEEE Press (1994)
7. Aurum, A., Wohlin, C.: The Fundamental Nature of Requirements Engineering Activities as a Decision-Making Process. Inform. Software Tech. 45, 945–954 (2003)
8. Rus, I., Lindvall, M.: Knowledge Management in Software Engineering. IEEE Soft. 2, 26–38 (2002)
9. Pohl, K., Dömges, R., Jarke, M.: Decision-Oriented Process Modelling. In: Software Process Workshop 1994, pp. 124–128. IEEE Press, Airlie (1994)
10. Mangan, P., Sadiq, S.: On Building Workflow Models for Flexible Processes. In: ADC 2002: Proceedings of the 13th Australasian Database Conference, pp. 103–109. Australian Computer Society, Darlinghurst (2002)
11. Sadiq, W., Orlowska, M.E.: On Capturing Process Requirements of Workflow-Based Business Information Systems. In: BIS 1999, pp. 281–294. Springer, London (1999)
12. Guindon, R.: Designing the Design Process: Exploiting Opportunistic Thoughts. Human-Compu. 5, 304–344 (1990)
13. Buckingham Shum, S.: Negotiating the Construction of Organizational Memories. In: Borghoff, U., Parechi, R. (eds.) Information Technology for Knowledge Management, pp. 55–78. Springer, Berlin (1998)
14. Wang, M., Wang, H.: From Process Logic to Business Logic—A Cognitive Approach to Business Process Management. Inform. Manage. 43, 179–193 (2006)
15. Kwan, M.M., Balasubramanian, P.R.: Dynamic Workflow Management: A Framework for Modeling Workflows. In: Proceedings of HICSS-30, pp. 367–376. IEEE Computer Society Press, Wailea (1997)
16. van Merriënboer, J.J.G.: Training Complex Cognitive Skills. Educational Technology Publications, Englewood Cliffs (1997)
17. Abecker, A., Dioudis, S., van Elst, L., Houy, C., Legal, M., Mentzas, G., Müller, S., Papavassiliou, G.: Enabling Workflow-Embedded OM Access with the DECOR Toolkit. In: Dieng-Kuntz, R., Matta, N. (eds.) Knowledge Management and Organizational Memories, pp. 63–74. Kluwer Academic Publishers, New York (2002)
18. Iivari, J., Hirschheim, R., Klein, H.K.: A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies. Inform. Syst. Res. 9, 164–193 (1998)
19. Zhuge, H.: Workflow- and Agent-Based Cognitive Flow Management for Distributed Team Cooperation. Inform. Manage. 40, 419–429 (2003)

20. Newell, A., Simon, H.A.: Human Problem Solving. Prentice-Hall, Englewood Cliffs (1972)
21. Hogarth, R.: Judgement and Choice, 2nd edn. Wiley, New York (1987)
22. Lehto, J., Marttiin, P.: Decision-Based Requirements Engineering Process. In: Workshop on Collaborative Embedded Systems Development, 6th International Conference on Product Focused Software Process Improvement, Profes. Springer, Heidelberg (2005)
23. Simon, H.A.: The Structure of Ill-Structured Problems. Artif. Intell. 4, 181–201 (1973)
24. Robillard, P.: The Role of Knowledge in Software Development. Commun. ACM 42, 87–92 (1999)
25. Jonassen, D.H.: Toward a Design Theory of Problem Solving. ETR&D-Educ. Tech. Res. 48, 63–85 (2000)
26. Gourgey, A.F.: Metacognition and Basic Skills Instruction. Instr. Sci. 26, 81–96 (1998)
27. Olewnik, A., Lewis, K.: A Decision Support Framework for Flexible System Design. J. Eng. Design 17, 75–97 (2006)
28. Hazelrigg, G.A.: A Framework for Decision-based Engineering Design. J. Mech. Design 120, 653 (1998)
29. Besharati, B., Azarm, S., Kannan, P.K.: A Decision Support System for Product Design Selection: A Generalized Purchase Modeling Approach. Decis. Support Syst. 42, 333–350 (2006)
30. Jin, Y., Lu, S.Y.: Agent-Based Negotiation for Collaborative Design Decision Making. CIRP Annals-Manuf. Techn. 53, 121–124 (2004)
31. Marston, M., Allen, J.K., Mistree, F.: The Decision Support Problem Technique: Integrating Descriptive and Normative Approaches in Decision-Based Design. Eng. Val. Cost Anal. 3, 107–129 (2000)
32. Cooper, R.G.: Winning at New Products. Kogan Page, London (1988)
33. Runeson, P., Höst, M.: Guidelines for Conducting and Reporting Case Study Research in Software Engineering. Empir. Softw. Eng. 14, 131–164 (2009)
34. Yin, R.K.: Case Study Research: Design and Methods. Sage Publications, Inc., Thousand Oaks (2009)
35. Kelanti, M., Hyysalo, J., Kuvaja, P., Oivo, M., Välimäki, A.: A Case Study of Requirements Management: Toward Transparency in Requirements Management Tools. In: Proceedings of the Eighth International Conference on Software Engineering Advances (ICSEA 2013), pp. 597–604. IARIA XPS Press (2013)
36. Berggren, E., Bernshteyn, R.: Organizational Transparency Drives Company Performance. J. Manage. Dev. 26, 411–417 (2007)
37. Simon, H.A.: Bounded Rationality and Organizational Learning. Organ. Sci. 2, 125–134 (1991)
38. Omoronyia, I., Ferguson, J., Roper, M., Wood, M.: A Review of Awareness in Distributed Collaborative Software Engineering. Softw. Pract. Exper. 40, 1107–1133 (2010)
39. Dourish, P., Bellotti, V.: Awareness and Coordination in a Shared Workspace. In: Proceedings of the ACM Conference on Computer-Supported Cooperative Work, pp. 107–114. ACM, New York (1992)
40. Robertson, T.: Cooperative Work and Lived Cognition: A Taxonomy of Embodied Interaction. In: Fifth European Conference on Computer-Supported Cooperative Work ECSCW 1997, pp. 205–220. Springer, Netherlands (1997)
41. Basili, V., Lindvall, M., Regardie, M., Seaman, C., Heidrich, J., Münch, J., Rombach, D., Trendowicz, A.: Bridging the Gap between Business Strategy and Software Development. In: Proc. International Conference on Information Systems, Montreal, Canada, pp. 1–16 (2007)

42. Molina, H., Olsina, L.: Towards the Support of Contextual Information to a Measurement and Evaluation Framework. In: 6th International Conference on the Quality of Information and Communications Technology, QUATIC 2007, pp. 154–166. IEEE, Washington, DC (2007)
43. Wasmer, A., Staub, G., Vroom, R.W.: An Industry Approach to Shared, Cross-Organisational Engineering Change Handling—The Road Towards Standards for Product Data Processing. Comput. Aided Design 43, 533–545 (2011)
44. Ruhe, G.: Software Engineering Decision Support – A New Paradigm for Learning Software Organizations. In: Henninger, S., Maurer, F. (eds.) LSO 2003. LNCS, vol. 2640, pp. 104–113. Springer, Heidelberg (2003)
45. Knight, J.C., Myers, E.: An Improved Inspection Technique. Commun. ACM 36, 51–61 (1993)

Automated User Interaction Analysis for Workflow-Based Web Portals

Emil Backlund[1], Mikael Bolle[2], Matthias Tichy[3], Helena Holmström Olsson[4], and Jan Bosch[3]

[1] ATEA and Chalmers University of Technology
emil.backlund@atea.com
[2] Chalmers University of Technology, Sweden
mikael@bolle.se
[3] Chalmers and University of Gothenburg, Sweden
matthias.tichy@cse.gu.se,jan.bosch@chalmers.se
[4] Malmö University, Sweden
helena.holmstrom.olsson@mah.se

Abstract. Success in the software market requires constant improvement of the software. These improvements however have to directly align with the needs of the users of the software. A recent trend in software engineering is to collect post-deployment data about how users use a software system. We report in this paper about a case study with an industrial partner in which (1) we identified which data has to be collected for a web-based portal system, (2) implemented the data collection, and (3) performed an experiment comparing the collected data with answers of the test subjects in a survey.

Keywords: user interaction, post-deployment data collection, Build-Measure-Learn, data-driven software engineering.

1 Introduction

Customer satisfaction is key for a software product to be successful. To achieve this, software development companies need to know what their customers value and how they interact with the product [1,2]. Without this knowledge, requirements prioritization becomes a challenging process in that product management has no accurate understanding of customer needs. In many companies, the feedback loop is slow and there are no efficient mechanisms that allow for continuous collection and analysis of customer data. If using "Lean Startup" terminology, there is no "Build-Measure-Learn" (BML) loop in place for continous validation of customer value [3]. In this loop, customer data serves as input for product management as well as for the entire development organization [3], and the feedback loop from customers is fast. Without the opportunity to continuously validate what customers value, there is the risk of lack of alignment between product and customer needs, as well as the risk of investing in R&D efforts without having an accurate way of continuously validating whether these efforts correspond to customer needs. What companies need are mechanisms that allow for continuous

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 148–162, 2014.

learning about customers, about product usage and about what functionality adds value to customers.

Today, this knowledge is typically gained by interacting with customers using techniques such as use cases, scenarios, prototypes, interviews. Also, alpha- and beta testing, observations, and customer surveys are efficient mechanisms for continuously validating that the software functionality that is developed is of value to the customers. During the early 2000's, agile software development methods gained traction in most software development companies as a way to increase customer and end-user interaction [4]. With an emphasis on close customer collaboration and small development teams, agile methods have proven successful for shortening feedback loops and reducing time-consuming coordination processes for a wide range of companies [5]. However, and experienced as problematic, most agile techniques for customer and end-user involvement assume face-to face interaction between developers and customers, something that is difficult to achieve in an increasingly global and distributed development environment. In most large-scale software development companies there is limited, or no, direct contact with customers or end-users, and it might require costly and complex procedures to gather and extract in-depth knowledge about how a specific product, or a specific feature, is used. Moreover, asking customers what they want is problematic. As recognized in research within human-computer interaction decades ago [6,7], the expected use of a system does not necessarily correspond to the actual use of that system. Often, there is a gap between what people say and what they do [8], which makes asking customers what they want a difficult task.

More recently, and due to the increase in connectivity and the on-line nature of products and systems, data can be collected as soon as customers use these. In Web 2.0 systems, and in on-line Software-as-a-Service (SaaS) technologies, automatic collection of customer data is the main source of input for learning about customers [9]. The cost of collecting data is low [10], and customer value can be validated on a continuous basis. Already, there is research indicating that automated data collection is conducive to an increased understanding of customer behavior and preferences [1,2].

However, while automated collection of customer data allows for an increased understanding of customer behavior – what is it we learn? Does what we learn about a product by automatically collecting data correlate with the actual customer perceptions of that product? And what is it that we learn from automatically collected data that we do not learn by asking customers?

In this paper, we explore how to automate the collection of customer data, and how analysis of this data helps companies increase their understanding of customers. In particular, we are interested in comparing different data sources, i.e., what companies learn from different data sources such as (1) asking customers how they perceive a system, and (2) automatically collecting data about how customers use a system. In our study, we investigate how automatically collected data compares with results gained in a customer survey, i.e., how data collected by automatically measuring product usage compares with test results

when asking customers how they perceive the product. Our research questions are the following:

1. RQ1: What knowledge about how customers interact with a system is interesting for developers, testers, project managers of a web-based portal software?
2. RQ2: How well do automated data collection and analysis methods compare with actual user perceptions?

The case for our research is a software solution, called Accelerator, developed by ATEA Global Services that automates the tasks of a Service Desk. The software enables employees to perform IT related tasks like ordering software and hardware. Each task resembles a workflow which is divided into a sequence of steps which need to be performed to finish the task. The goal of ATEA Global Services is to automate as many of these tasks as possible. To achieve this, accessibility and understanding of end-user needs is essential. Therefore, ATEA Global Services wants to understand how these users interact with their software. The product that ATEA Global Services provides consists of thousands of lines of code, which makes it difficult and time consuming for the developers to implement a monitoring solution. On the other hand, a survey can also be difficult as they do not have direct contact with their end-users.

The contribution of this paper twofold. First, we present a list of knowledge needs about how user interact with a workflow based web system. Second, we developed an automated data collection method and compared the results in a user experiment with the users feedback in an online survey. As a result, we got a weak correlation between the answers of the users in the web survey and the results of the automated data collection.

In the next section, we discuss in more detail the needs for a valid automated user behavior analysis as part of the Build-Measure-Learn loop to enable companies consistently evaluate and improve their systems. Section 3 describes our research method to answer the presented research questions. The answer to research question 1 were identified by a workshop with developers, testers, and project managers at ATEA Global Services. The workshop and its results are described in Section 4. Section 5 presents the automated approach to collect user interactions to satisfy the knowledge needs identified before. In Section 6, we present the experiment to answer research question 2 how well the automated approach compares with an online survey with users. After a discussion of related work in Section 7, we conclude and give an outlook on future work.

2 Background

To increase customer satisfaction and to add customer value is key to any software development company in order to stay competitive [11]. Typically, ideas for improvement and innovation of features are collected and prioritized during the early phases of road mapping and requirements engineering, and as part of a planned product release cycle [12]. The selection of what ideas to include is

done by product management and forms the basis for enormous R&D investments. What has shown problematic is how to continuously confirm the correctness of the decisions taken during the requirements prioritization process. Often, product management has no accurate way to continuously validate whether the features they prioritize are those that add value to customers [1,2]. As a result, requirements prioritization becomes a challenging process in which companies run the risk of having opinions inform decision-making rather than data reflecting actual customer usage. Moreover, and as recognized in human-computer interaction (HCI) research decades ago, asking customers what they want is difficult since what customers say does not necessarily reflect what they do [6]. In using well-established managerial behavioral terminology, espoused behavior is often different from the theory-in-use [8], meaning that there is a difference between what people say and what they do. To address this challenge, and to learn from customer usage rather than from their opinions, companies need to find mechnisms that allow them to continuously collect customer data. Also, they need to find ways in which to analyze and understand this data in order to understand what they learn from different data sources.

Below, we outline the Build-Measure-Learn (BML) loop which is a central concept within the lean startup community [3]. The concept is relevant in that it emphazises the importance of customer feedback, as well as the continuous need for this. Furthermore, we introduce the concept of data-driven software engineering. In relation to our research, both these are good examples of practices where continuous collection of customer data steer adjustments and decision-making. Also, these practices reflect the sincere interest in the software engineering field to find mechanisms that help validation of success in terms of customer value.

2.1 The Build-Measure-Learn Loop

As a central concept within the lean startup community, there is the "Build-Measure-Learn" (BML) loop, which is described as the concept of validated learning [3]. In this loop, ideas are quickly turned into testable products, data is gathered by measuring how the product is actually used by a selected group of customers, and ideas for product improvement and innovation are based on what is learned by analyzing the data collected from customers. In this way, focus is always on developing and delivering customer value, and the model advocates an approach in which continuous customer validation is critical. The concept was developed by Ries [3] after noticing that the solution focused thinking that characterizes the agile development approaches often leads to a situation in which many software companies fail. What he found was that instead of continuously evaluating what customers value, most companies spend time and money developing products without knowing whether customers will be interested or not. While projects are delivered on time and on budget, there is the risk that nobody wants the product. In the BML loop, the main intention is to emphasize the importance of continuous validation with customers in order to understand the problem during product development and improvement.

Besides the BML loop, another central concept within the lean startup community is the "pivot", a term used when a company changes direction based on what they learn from customers, i.e., from customer data. Ries [3] claims that having "pivoted" is the most frequently occurring commonality among successful startups, and that successful companies seldom end up doing what they initially set out to do. Rather, they change direction based on efficient collection and analysis of customer data and what they learn from customers.

2.2 Data-Driven Software Engineering

Data-driven software engineering is a practice in which continuous collection of data is used to understand the successful development of software systems [13]. During the development cycle, different metrics related to product quality are collected, and the goal is to use such metrics to make estimates of post-release failures early in the software development cycle, as well as during the implementation and testing phases. Such estimates can help focus testing, code and design reviews and affordably guide corrective actions and decision-making activities.

One area in which data-driven software engineering has been successfully applied is the area of Test-Driven Development (TDD). Test-driven development [14] is an "opportunistic" software development practice that has recently re-emerged as a critical enabling practice of agile software development methodologies after having been used sporadically for decades [14,15]. With this practice, a software engineer cycles minute-by-minute between writing failing unit tests and writing implementation code to pass those tests. By using the data-driven software engineering approach, previous studies have investigated whether test-driven development actually work, and if so, if there is supporting data for development teams to make informed decisions during product development [14].

In summary, and in relation to this research, the BML loop and the concept of data-driven software engineering are both good examples of practices where continuous collection of customer data work as a basis for product development and improvement. These practices reflect the interest in the software engineering field to find mechanisms that help automatic collection of metrics, and as a result, continuous validation of customer value.

However, while there are numerous mechanisms for automatic collection of customer data, we are interested in exploring how well this data also compares with the perception held by customers. In our study, we investigate if there is a link between raw data and the assessment made by customers when asking them how they perceive a system. In the following sections, we outline our research method and the results from our case study including the results from a comparison between automated data collection results and customer survey results.

3 Method

In the following, we describe our research methodology as sequence of six steps as shown in Figure 1: (1) The understanding of the case and its challenges, (2) conducting a workshop to understand what should be analyzed captured as questions about user interactions, (3) breaking down the outcome from the workshop into a set of questions answering research question RQ1, (4) implementing an automated data collection for the system, (5) performing user tests including a web survey to generate data, and (6) analyzing the results from step four and five to answer research question RQ2. The fourth and fifth step ran in parallel because of their mutual dependency. Each step is discussed further beneath.

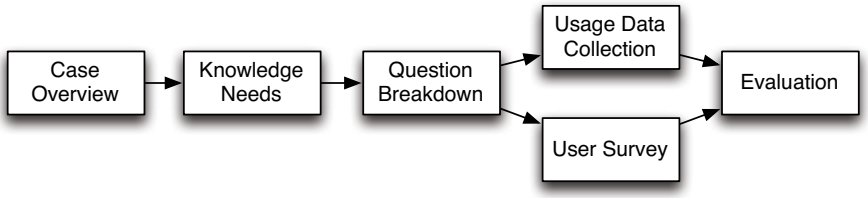

Fig. 1. Research method

To understand the case a brief introduction of the system was held by ATEA Global Services. Also, access to the system and the source code was provided for in depth analysis. Beside this, there were both formal and informal meetings conducted with different employees within the organization. Based on this information insight and general understanding of the case was gained.

Then, a workshop was conducted with employees of different positions at ATEA Global Services. The main goal of the workshop was to gain more insight of their perception of the system as well as identifying needs for knowledge about user interactions. The needs were formulated as questions about how users interact with the system.

Since the list of questions where not detailed enough for an automated data collection, they were broken down into more detail and made unambiguous, see Section 4.2. At this stage, it was also determined which questions could be answered by the automated data collection respectively by conducting a web survey.

To answer research question RQ2, we let test subjects perform a set of tasks and then let the them answer questions in a web survey, this is annotated in Figure 1 as User Survey. The tasks performed were related to a set of questions extracted from the workshop. The survey, see Section 6, was designed by examining the extracted questions from the workshop and their breakdown. Only a subset of the questions could be evaluated since some questions required that

the data collection has been actively used for some time. With the questions for the survey selected, tasks were designed that would be sufficient to answer those questions by the automated data collection. When the tasks had been designed the testing was executed by inviting users with similar profile as possible end-users. They were given an introduction of the case product, before performing the tasks, and afterward they were given access to a web survey. By having this approach it was possible to simulate a real world scenario. As previously mentioned, data collection of usage data was implemented and tested in parallel to the User Testing, see Section 5.

To asses the validity of the approach a comparison between results from data collection of data from tasks and result from the web survey was done, see Section 6. By doing this it was possible to evaluate how well an automated data collection method compares with a web survey, which is conducted to understand user interaction. This is final step in our method. Due to space restrictions, we cannot present the full results and refer the interested reader to [16].

4 Identification of Knowledge Needs

To identify the company employees' point of view on their need of customer understanding a workshop was conducted. Invited to the workshop were parts of the support team, test team, development team, and the product owner. This set up of participants was selected to get a wide range of ideas and to enhance discussion, as people from different teams are very likely to have different standpoints.

The goal of the workshop was to gain more insight of participants perception of the system and their ideas on knowledge needs about user interaction with the system that could be used as a foundation for defining the questions that should be answered by the automated data collection.

The workshop had seven participants from the company. It started with a brief introduction to the idea of this study and the goal and structure of the workshop, this lasted for about ten minutes.

The participant were then presented with questions, one at a time, which all were first described for about two minutes so that everyone understood the question. After a question was introduced each participant wrote down his or her ideas to the question on sticky-notes, which were collected and put on a whiteboard. The time limit, which participants had while writing down their ideas, was strictly five minutes. The reason of not letting them work in groups, at this point, was to avoid them from influencing each other and decreasing productivity.

The notes that were similar to each other were then grouped, and simultaneously there was an open discussion about each group of notes. The reason for this structure during the workshop was to first let people think alone and then to have an open discussion to reason about their answers and see if more could be extracted, each open discussion had a time limit of ten minutes. By grouping the questions it was easier to have a structured discussion and see which points seemed to be common perceptions between participants.

In total, for all questions, 74 sticky-notes were written. The complete workflow of the workshop session can be seen in Figure 2. The questions for the workshop were designed in such a way that they together would give a deeper knowledge of what was known, what was not known, and what was interesting to know. This knowledge could then be used to define a number of questions that should be answered by the survey and the data collection.

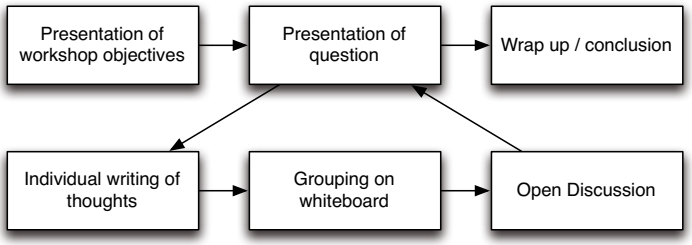

Fig. 2. Workflow of the workshop process

In the following, we review two questions of the workshop and the results. The third question dealt with porting the software to a different platform which is out of scope for this paper.

Which Parts of the Software Need Improved User Experience? The purpose of this question was to understand what the employees at ATEA Global Service saw as points of improvement in regards to the user experience. By questioning them it would be possible to predict the outcome of the gathered data and understand if they are able to predict the need of their end user. This question was the foundation for defining which parts of the software were important to monitor and analyze. The sticky-notes were gathered and grouped into five different categories as shown in Table 1. The diversity of participants was very important for the discussion of each category and it resulted in a clear view of possible parts to monitor and analyze.

What Would You Like to Know About the Users' Interaction with Accelerator? By asking this question, it was possible to find points where the employees of ATEA Global Services felt uncertain in regards to how the software is used. The idea was also to understand if parts of the software had been developed and maintained without clear and motivated reasons. The question was defined as guiding purpose to which data that should be gathered and extracted for analysis. The categories shown in Table 2 have been identified.

From the open discussion, it was clear that participants where uncertain of how the product was being used. Which seems to come from a lack of collaboration with the actual end users at the customer companies. It was also clear

Table 1. Question categories on required improved user experience

What happens?	The transparency of the software towards the end-user. Example of sticky-notes: "Orders: What happens with the order when you complete your order."
Grouping	The possibility to group and simplify some procedures. Example of sticky-notes: "The need to go to different parts of the system to order different types of products."
Admin	The parts that are related to the Admin interface and advanced configuration.
Menu	The ideas discussing the navigation. Example of sticky-notes: "Menu system, there is a limited space"
Customization	The reasoning about a more personalized interface. Example of sticky-notes: "An Accelerator for different users' roles."

Table 2. Question categories on knowledge required about user interaction

Misunderstandings	See where end user have issues to use the product. Example of sticky-notes: "What makes a user confused, regards to how the software works"
Statistics	Information based on descriptive statistics. Example of sticky-notes: "Least used part of the software"
Time	Information based on time, when and how end user use the software. Example of sticky-notes: "How long time does it take a user to complete different tasks"
Frequency	How often and how is the software used. Example of sticky-notes: "How big is the user group of daily users"

that the participants found this question interesting and wanted to know more about their end users, as there were so many different ideas. The result from this questions was in the following used to define what data to gather.

4.1 Identified Data Analysis Needs

There were some obvious outcomes from the workshop, first of all there is a need for increased communication within the organization. It was also clear that many felt uncertain about their end users usage of the software. This emphasizes the need of more statistical data of the software use or increased communication with end users.

By gathering and analyzing the information from all Workshop Questions the following eleven questions were extracted and defined to possibly be answered by the survey and the automatic data collection:

1) What is the frequency of use for functionality: By answering this question usage and possibly user interface design decisions can be made better. It defines which parts should be prioritized when evolving the product.

2) Frequency of use for different roles: The reason for understanding frequency of use for different role is so that the user interface could be tailored for some roles which use the system a lot.

3) Are there any functions that are not used: The question aims at reducing complexity by removing not used functionality.

4) How is the internal search of the website utilized: Since many parts of the software are based on search it is interesting to know if the search functionality is used once within a session or several times. If search mostly occurs once it might be possible to suggest a simplified design. Furthermore, the system supports a test based search and a tree based search.

5) How long time does it take a user to complete a task: If a part of the software is complex, it most probably will take longer time to complete task. By understanding complex parts, it would be possible to pinpoint where to focus on improvement or parts that should be revised.

6) How does software use differ between regular and non-regular users: Based on this question, strategic decision can be made from the data. For example, which type of user should be prioritized.

7) Which task is the most difficult to complete: By understanding what scenario and part of navigation that are difficult for the end-user, it is possible to see where it is most vital to simplify the user interface.

8) Are there any trends, between different versions: Changes over time are important feedback for continues development and enables an evaluation of previous decisions as suggested in the BLM loop.

9) Are there parts of the software with a high bounce rate: By finding parts of the software with a high bounce rate it is possible to revise them. A bounce is when a user navigates to a certain part and then navigates to another one in just a matter of second. This indicates that the former was not what the user was looking for. A high bounce rate can be a sign of design or content issues.

10) How often and where do drop offs occur: A drop off occurs when a user works on a task and then aborts. By finding drop-offs, it could be possible to pinpoint parts that have a complexity that is too high or where there are too many steps and options for the end-user to complete.

11) What time of the day are tasks carried out: By understanding when the task is started and what is carried out, it might be possible to motivate what parts should be extracted to a handheld device as these may be used more often in the mornings, evenings or during lunch.

4.2 Question Refinement

In the following, we describe how the questions defined in Section 4.1 are redefined in a more detailed manner and broken down into a tree structure. Moreover, the questions are analyzed to identify which could be answered by automated data collection and respectively a web survey.

Each question has been broken down further and annotated by notes which indicate whether an answer to a question is feasible by using the automated data collection or by using a survey. Typically, the workshop questions deal with

user impressions and are therefore not feasible to answer by an automated data collection approach but with an online survey whereas the broken down questions are on a level which can be answered by automated data collection but maybe not with a survey. We use «Survey» for questions answerable by a survey and those answerable by automated data collection with «DataCollection». We focus in the following on the workshop question *7) Which task is the most difficult to complete* and refer to [16] for the full description of all breakdown questions.

Breakdown Question 7 is the same as question 7 from the workshop as presented in section 4.1. For other workshop questions this might be different. The purpose of this structure is that a survey could answer Breakdown Question 7 directly by asking the participants how difficult each task is. However, to design a survey question for the two refined questions, 7.1 and 7.2, would be to complex. These two questions try to identify which task are difficult to complete by analyzing the longest time it takes a user to complete a step during the completion of a workflow, represented as question 7.1, and by looking at the task which has the highest value of completion time per steps, represented in question 7.2. Both can be answered automatically from the collected data.

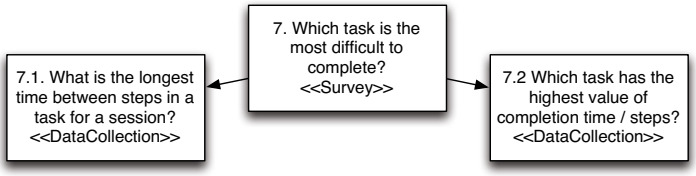

Fig. 3. Breakdown Question 7

As an outcome of this activity the initial 11 questions have been refined into 36 refined questions and for each question, it was defined whether they can be answered by an automated data collection or a web survey.

5 Automated Data Collection

The user interaction is collected as a set of traces, where a trace is a record of a single user interaction with the system.

The process of developing the data collection included three steps: First, determining what data needs to be collected. Second, selecting an appropriate technology for implementing usage tracing. Finally, executing the implementation of usage tracing with the selected technology to capture the user interaction and to answer the questions presented in Section 4.2. Each stored user interaction trace includes which part of the system is used, who is the user, what is the user's action (e.g., a button click), the roles of the user, and time related information like timestamp and execution time.

As the user interactions are handled by a wide variety of source files, the addition of data collection would require many manual changes. However, as tracing is a true cross cutting feature and the source code closely followed guidelines about naming of methods, aspect oriented programming [17] was used to introduce the usage tracing code into the system. We used 11 point cuts to cover the complete usage tracing by introducing the code into all methods which dealt with user interactions in the system like all onClick-methods, which are called every time a user clicks on an user interface element. We were able to exploit company specific source code naming conventions.

The usage of aspect oriented programming enabled a fast development of the data collection code. Furthermore, it supports the automatic evolution of the usage tracing in case new functionality is added which follows the same guidelines as the existing code.

6 Preliminary Evaluation

This section describes a preliminary evaluation that the solution in section 5 produces knowledge that conforms to the users perception expressed in a web survey. The evaluation was performed by letting 15 test subjects take part in a user testing workshop. In the workshop, each user was asked to perform the set of tasks shown in Table 3.

Table 3. Tasks executed by the test subjects in a random order

1. Request Hardware	User ought to complete a request for a computer.
2. Change Password	User ought to successfully change the password.
3. Request Software	User ought to complete a request for a software product.
4. Request Access	User should request membership to a user group.
5. Approve Order	User should approve a request, while using a manager account.
6. Cancel Order	User should cancel a request.

These six tasks cover the essentials of the case product and what users most often use the product for according to ATEA Global Services. Since requests of different kinds are the idea of the case product three tasks were used to cover this. Requests need approval in some cases and due to this the Approve Order and Cancel Order were selected. However, due to issues with the automated data collection, these two tasks could not be measured and thus are excluded from the following description. To also cover some of the manage user part of the product one task was designed for changing password.

As mentioned in Section 3 the tasks were designed to reflect a real world scenario. With this in mind the task were formulated in such way that they represent a possible scenario that end-users could encounter. To counter undesired variations related to learning- and boredom-effect the ordering of tasks for each test subject was made so that no subjects performed the tasks in the same order.

After completion of the tasks, the users were sent an e-mail with a link to a web survey. After the data collection had been run on the user testing data

and the data from the web survey had been collected, the results were compared. Four Breakdown Questions were selected, number 1 (Time for a user to complete a task), 3 (Product bounce rate), 5 (Utilization of internal search) and 7 (Most difficult task to complete). We will report about the last one and refer to [16] for the others. The web survey question related to this Breakdown Question was: "Rate how difficult each task was to complete on a scale from 1 (easy) to 4 (very difficult)."

The most difficult task was Change Password which was rated 2 on average. Then came Request Access with 1.72 followed by Order Hardware and Order Software with 1.52 and 1.3.

The corresponding measurements are the average completion time per step for each task. The highest average completion time per step by far was 47.4 seconds for the Change Password task. Then came Order Hardware (14.9 seconds) and Order Software (9.0 seconds). Finally, Request Access had an average completion time per step of 8.3 seconds. These three tasks have a similar small average completion time per step in comparison with Manage Account.

When we compare the average completion time per step with the survey results, we see both the measurement as well as the survey indicate that Change Password is the most difficult task. This was also seen during the experiment as the test subjects struggled choosing a password which conforms to the password rules which were enforced but not explained by the system. Request Access was rated higher in the survey than indicated in the measurements. The ranking between Order Hardware and Order Software was the same in measurements and the survey. Kendall's tau is 0.333 indicating a weak positive correlation. We refer to [16] for more details about the other parts of the evaluation including results of statistical tests.

The measurements for the three other tasks than Change Password are very similar. Thus, small changes in the measurements may lead to different ranks and also the users might not be able to judge the small differences. So, we plan to do follow-up evaluations with more subjects.

7 Related Work

Automated user interaction analysis is well known in the web development with Google Analytics as one of the major platforms. As an example, Hasan et al. report about using Google Analytics for e-commerce systems [18]. Google analytics and similar systems, however, have the problems that the data collection and analysis is done on systems not owned by the company which raises privacy issues. Furthermore, the data collection is constrained by the supported features.

Van der Shuur and Jansen [19] have presented a solution for improving software quality by automatically gathering and reporting how a software service is being used. The data gathering was implemented in the service layer using aspect-oriented programing as in our approach. Furthermore, the reporting was built using a set of metrics that were concerned with quality attributes like availability, accuracy, reliability and usability. They concluded that their solution was

expected to contribute to an increase in software quality and that future work was needed on how to use data mining techniques for reporting on software utilization.

For usage tracing aspect-oriented programming stands out as having been tested for its suitability when implementing automated data collection for usability evaluation and usage tracing. Tart and Moldovan showed that it could be used for automated usability evaluation [20] and equal results where gained by Tao who used the same framework [21]. Tarby et al. compared aspect-oriented programming with Agent-Based Software Architecture concluding that they could be used as complement. The recommendation was to use aspect-oriented programming for defining traces while the agents would be used to "produce traces whose visualization will be made in real time" [22]. A trace is referred to as a record of an action performed by a user.

However, these papers lack a comparison against other data collection techniques compared to this paper.

8 Conclusion and Future Work

The presented work is concerned with enabling software companies to measure and subsequently learn how their customers use their software as a precondition for improving the software in such a way that it benefits the customer. We used a web-based portal software developed by ATEA Global Services as an industrial case.

The different aspects, which the company were interested to know about how user interact with the software, were identified in a workshop with different roles. An automated data collection system for the identified aspects was built and in a follow-up experiment the results of the automated data collection system was compared to answers of the test subject in a survey. We refer to [16] for the complete results and further details of our research study.

Future works include a follow-up experiment with a higher number of test subjects as well as monitoring the system and its measurements over several versions to identify whether the added automated data collection capabilities enable the company to continually improve the software.

References

1. Holmström Olsson, H., Bosch, J.: Towards data-driven product development: A multiple case study on post-deployment data usage in software-intensive embedded systems. In: Fitzgerald, B., Conboy, K., Power, K., Valerdi, R., Morgan, L., Stol, K.-J. (eds.) LESS 2013. LNBIP, vol. 167, pp. 152–164. Springer, Heidelberg (2013)
2. Holmström Olsson, H., Bosch, J.: Post-deployment data collection in software-intensive embedded products. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 79–89. Springer, Heidelberg (2013)
3. Ries, E.: The Lean Startup: How Constant Innovation Creates Radically Successful Businesses. Penguin Group, London (2011)

4. Highsmith, J., Cockburn, A.: Agile software development: The business of innovation. IEEE Computer 34(9), 120–122 (2001)
5. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J.: New directions on agile methods: A comparative analysis. In: Clarke, L.A., Dillon, L., Tichy, W.F. (eds.) ICSE, pp. 244–254. IEEE Computer Society (2003)
6. Morris, M., Dillon, A.: How user perceptions influence software use. IEEE Software14(4), 58–65 (1997)
7. Soloway, E., Guzdial, M., Hay, K.E.: Learner-centered design: the challenge for hci in the 21st century. Interactions 1(2), 36–48 (1994)
8. Argyris, C., Schön, D.: Organisational learning: A theory of action perspective. Addison Wesley (1978)
9. Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M.: Controlled experiments on the web: survey and practice guide. Data Mining and Knowledge Discovery 18(1), 140–181 (2009)
10. Bosch, J.: Building products as innovation experiment systems. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 27–39. Springer, Heidelberg (2012)
11. Dzamashvili-Fogelström, N., Gorschek, T., Svahnberg, M., Olsson, P.: The impact of agile principles on market-driven software product development. Journal of Software Maintenance 22(1), 53–80 (2010)
12. Sommerville, I.: Software Engineering, 6th edn. Pearson Education, Essex (2001)
13. Bird, C., Murphy, B., Nagappan, N., Zimmermann, T.: Empirical software engineering at microsoft research. In: Hinds, P.J., Tang, J.C., Wang, J., Bardram, J.E., Ducheneaut, N. (eds.) CSCW, pp. 143–150. ACM (2011)
14. Beck, K.: Test Driven Development: By Example. Addison-Wesley Professional (2002)
15. Cockburn, A.: Agile Software Development. Addison-Wesley Professional (2001)
16. Backlund, E., Bolle, M.: Automated usage tracing and analysis: a comparison with web survey. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden (2013)
17. Kiczales, G., Lamping, J., Mendhekar, A.: Aspect-oriented programming. In: Akşit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
18. Hasan, L., Morris, A., Probets, S.G.: Using google analytics to evaluate the usability of e-commerce sites. In: Kurosu, M. (ed.) HCD 2009. LNCS, vol. 5619, pp. 697–706. Springer, Heidelberg (2009)
19. van der Schuur, H., Jansen, S., Brinkkemper, S.: Becoming responsive to service usage and performance changes by applying service feedback metrics to software maintenance. In: Proc. of the 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops, pp. 53–62. IEEE (September 2008)
20. Tarta, A., Moldovan, G.: Automatic usability evaluation using aop. In: 2006 IEEE International Conference on Automation, Quality and Testing, Robotics, vol. 2, pp. 84–89. IEEE (2006)
21. Tao, Y.: Capturing user interface events with aspects. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4553, pp. 1170–1179. Springer, Heidelberg (2007)
22. Tarby, J., Ezzedine, H., Kolski, C.: Trace-Based Usability Evaluation Using Aspect-Oriented Programming and Agent-Based Software Architecture. In: Human-Centered Software Engineering, pp. 257–276 (2009)

Orchestrate Your Platform: Architectural Challenges for Different Types of Ecosystems for Mobile Devices

Herman Hartmann[1] and Jan Bosch[2]

[1] University of Groningen, Groningen, The Netherlands
j.h.hartmann@rug.nl
[2] Chalmers University of Technology, Gothenburg, Sweden
jan@janbosch.com

Abstract. The introduction of smartphones and tablets has led to a fast growing industry in which most firms have started an ecosystem-centric approach. In this paper three types of ecosystems are identified: Vertically integrated hardware/software platforms, closed source software platforms and open source software platforms. These ecosystems differ in the scope of the platform, i.e. covering both hardware and software, and the technology design, i.e. whether the software can be altered by the complementors. In this paper the challenges for each type of ecosystems are identified from an architectural point of view. Platform leaders can use our analysis to orchestrate their platform by proactively addressing the challenges that we identify and properly evolving the scope and technology design of their platforms.

Keywords: Platform leadership, mobile ecosystems, consumer electronics, software architectures, embedded systems.

1 Introduction

The last decade has seen a revolution in the consumer electronics industry in the wake of the introduction of smartphones and tablets. This industry consists of hardware and software vendors, handset makers and application developers [1]. Furthermore, smartphones have caused a renewed interest in tablets that are often replacing personal computers as the consumers preferred computing platform.

Initially this industry was dominated by a handful of firms that developed the hardware and software and created the handsets, e.g. Nokia, Siemens, Motorola and RIM. In a later stadium newcomers, such as Apple, entered the market because hardware platforms became commodities. Existing software companies entered the market, e.g. Microsoft and Google, and created a software platform that acts as a spanning layer between the applications and the hardware platform [2].

Due to the increasing development effort, many firms have adopted an ecosystem centric approach [3]. In this paper we identify the three types of ecosystems that have emerged. These ecosystems differ in the scope of the platform and whether the

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 163–178, 2014.

complementors can alter and contribute to the platform. We analyze these types with the framework of Gawer and Cusumano [4] by using a number of factors that influence the optimal scope and technology design. These factors cover general competitive criteria, design challenges that originate from the nature of embedded consumer electronic devices and factors that originate from the use of ecosystems. For each type of ecosystem we identify which factors are challenging, from an architectural point of view, and describe how the actors mitigate these challenges.

The key contributions of this paper are:

- Three types of ecosystems are identified and the factors that determine the optimal scope and technology design of the platform.
- Each type of ecosystem is evaluated and the factors are identified which are challenging for each type of ecosystem, from an architectural point of view.
- Case studies are presented that show how the actors address these challenges.
- An analysis of the market share is presented with future scenarios.

Platform leaders can use our analysis to orchestrate their platform by proactively addressing the challenges that we identify and properly evolving the scope and technology design of their platforms. The research in this paper is both descriptive, since it provides a classification of the existing situation, and explanatory because it describes the conditions under which the different types of ecosystem prosper.

The remainder of this paper is structured as follows: Section 2 provides background and Section 3 describes the type of ecosystems. Section 4 presents the factors and the evaluation for each type of ecosystem. In section 5 cases studies are presented, while section 6 gives a historical perspective and presents future scenarios. This paper ends with a comparison with related art in Section 7 and by our conclusions and recommendations for further research in section 8.

2 Background

Smartphones and tablets have characteristics of embedded devices as well as general purpose computing devices [1]. As an embedded device they are aimed to perform dedicated functions, such as making telephone calls, recording videos and playing music, which have real-time performance requirements. As a mobile device they have constrained computing resources and should use as little energy as possible because they operate on batteries and have few options for heat dissipation. These devices also have characteristics of general purpose computing devices because mobile phones and tablets are meant for a variety of tasks such as Internet browsing, reading and writing documents, accessing social media and playing games.

As an example look at Flash, an Internet browser plugin for watching videos. Apple has not allowed the use of Flash on their devices because it causes a significant shortened battery life [5], but has promoted the use of the H.264 video encoding standard for which their device is optimized. This example shows the conflicting requirements between a dedicated embedded device and a general purpose device.

The speed of innovation occurred at a high pace and the commercial lifetime of a product is often less than two years and therefore do not require backwards compatibility. A firm that is lagging behind may lose a market share rapidly.

2.1 System Architecture

In figure 1 a high level architecture of smartphones and tablets is presented, showing some of the actors. The architecture consists of a hardware platform an OS-kernel and middleware that offers a framework for application developers [6].

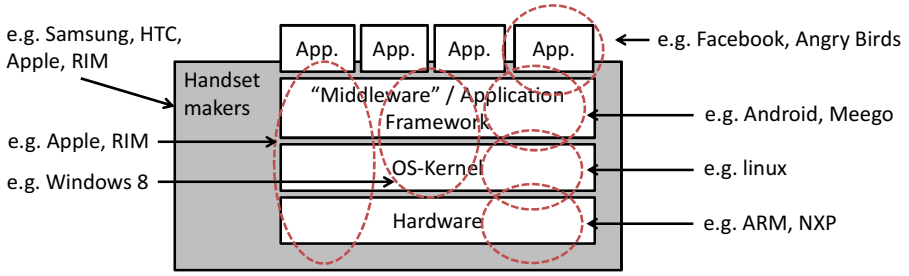

Fig. 1. System architecture of mobile phones and tablets with some actors

The hardware layer consists of System on Chips (SoCs) with peripheral Integrated Circuits (ICs). A SoC contains several dedicated building blocks on a single IC such as audio decoding, memory and a configured CPU. A dedicated SoC, which is common for embedded systems, is better able to meet the performance requirements at lower power consumption than using a general purpose CPU and separate ICs, as in the PC industry. The design of the software and the SoC are tightly coupled, e.g. for controlling the power consumption of separate hardware building blocks.

3 Growing Software Size: Move towards Ecosystems

In an earlier paper we analyzed the transition in the mobile phone industry from 1990 until 2010 [1]. Initially the mobile phones were developed by a small group of vertically integrated companies that developed the hardware, firmware and applications as well as created the handsets, e.g. Nokia, Siemens, Ericsson and Motorola.

Smartphones introduced the need for dedicated application processors, operating systems and software applications. The development investments, both for hardware and software, became significantly higher and therefore the companies needed to focus their activities. The mobile phone companies made their software platforms available to external parties. In this way these organizations transitioned from software product line engineering in an intra-organizational context to software ecosystems [3].

Separate hardware platform suppliers were created as spin-offs of the vertically integrated companies, such as Qualcomm [7] and Freescale. Some of these suppliers focused on a hardware platform only while others also integrated software platforms on

these platforms. When hardware platforms and other components become commodities it was possible for newcomers to enter the market without large investments. Examples of these newcomers are TCL, HTC and Apple.

Several software platforms entered the market and firms were aiming to create a common software platform in an attempt to replicate the Wintel model and create an ecosystem [8]. Similarly as the case with Microsoft Windows, a firm controlling the spanning layer can earn most of the revenues since it controls the interface towards the hardware as well as towards the applications [10, 2]. The number of competing mobile platforms that entered the market was huge, e.g. WebOS, Android, LiMo, Symbian, Windows Mobile, MeeGo etc. Until 2009 most of these attempts didn't gain industry wide adoption and therefore the industry was highly fragmented.

Starting in 2010 many of the vertically integrated firms lost market share and Android, an open source software platform, gained a share rapidly. Furthermore the dominant player in the PC market, Microsoft, increased their efforts to gain a dominant position. As a result in 2013 there is a fierce battle between different ecosystems.

3.1 Classification of Ecosystems with Their Complementors

This paper focusses on the ecosystems that are centered around the software platforms and the vertically integrated firms. For a comparison of these ecosystems we use a classification that is based on two properties: (1) is the software proprietary or open source, and (2) are the hardware and handset included, see figure 2.

	Proprietary closed Source	Open Source
Software Platform only	**(2)** Windows WebOS (2010-2012)	**(3)** Android, Tizen, Firefox OS
Hardware and handset included	**(1)** Apple, RIM Nokia (< 2013)	**(4)** Not present in the market

Fig. 2. Classification of ecosystem types

This classification results in four possible ecosystems:

1: Vertically integrated proprietary hardware/software platform. This platform consists of the hardware, proprietary closed source software and includes the handset. The complementors are the App developers. Examples of platforms and their leaders are Apple with the IPhone, Rim with Blackberry and Nokia with Asha.

2: Proprietary, closed source software platform. This platform consists of a proprietary closed source software platform. The complementors are the suppliers of hardware platforms, system integrators, handset makers and app developers. Examples of such platforms are Windows Phone and WebOS (2010-2012).

3: Open source software platform. This platform consists of an open source software platform. The open source software platform is based on the concept that multiple parties can contribute to share development effort and since the source is open, the handset makers can change, add or remove functionality. The complementors are the

suppliers of hardware platforms, system integrators, handset makers and app developers. Examples are Android, Tizen and Firefox OS.

4: Open source software and hardware platform. This platform would consist on an open source software platform including the hardware and handset. The users of such a platform, or better handset, would be able to change the code of the software platform and add their own functionality. We have excluded this type from this paper as this type, to our knowledge, is not available in the market.

Note that in the past some handset makers used an open source platform of which they acted as the orchestrator or platform leader, e.g. Nokia that used Symbian when it was open source and Motorola of Google that used Android. In this situation the handset makers are in-house complementors of the open source software platform and therefore the combination is not a separate type of ecosystem.

4 Defining the Platform Scope

Gawer and Cusumano designed a framework for a company to become a platform leader. A platform leader is driving the innovation in the industry, ensures the integrity of the overall system and creates opportunities for firms that create complementary products [4]. This framework includes (1) how a company defines the scope of the platform, i.e. what part of the product is developed in-house and what it leaves for complementary firms, (2) the technology design, i.e. to what degree the platform should be open to the complementors [12]. The scope and technology design highly determine the success of a platform. It should support the capabilities of the platform leader, allow for complementors to contribute and be able to respond to a changing environment. Therefore the scope and design of a platform evolve over time [4].

4.1 Factors that Influence the Optimal Scope of the Platform

For evaluating the scope and technology design, we use the following factors, each of which will be elaborated in more detail in the remainder of this section:

Cost, Quality, Flexibility, Innovation and **Variability**: These factors originate from the analyses of Bolwijn and Kumpe [13], who identified the main competitive forces that determine the success of a consumer electronics firms. In a later analysis Sheate [14] identified variability as an additional criterion, related to flexibility, since individual users increasingly have specific demands.

Efficient use of system resources and **Hard real time requirements:** These factors originate from the nature of embedded systems and consumer products [15].

Stability of the interface and **Effort for system integration**: These factors are specific for ecosystems since complementors have to rely on a stable and, ideally, backwards compatible interface [2], and for the set-makers the integration of the hardware and software is a substantial proportion of the development effort.

Other factors that determine the success of an ecosystem, such as business models and actions [16, 12] are not covered since this paper focusses on architectural aspects.

4.2 Evaluation of the Factors

In this section we analyze each type of eco-system using the factors from the previous section. Here we will "score" each factor on whether it is particular challenging or easier to address. Here + means: easier to address, 0 means: neutral, and - means: particularly challenging. The "scores" do not represent absolute values, but are solely used to express the differences between the types of ecosystems. This analysis is based on the experience of the authors, the information in literature and the case studies that are presented in section 5.

A: Costs. The larger the scope of the platform, the more development and maintenance effort is required by the platform owner. When multiple parties are involved these costs can be shared between the participants [17]. Because hardware and software development is costly, this is the main reason why ecosystems became widely adopted. Firms that aim for low costs specialize on a few products so that tasks become routine [13]. Consequently products that are developed in consort by specialized firms, especially when interfaces are pre-defined, can be made at lower costs.

For the vertically integrated type this factor is particularly challenging since they develop the entire platform including both hardware and software and can only amortize their investments over their own handsets (hence score = -). The open source platform can more easily address this, especially when the complementors contribute to the platform and often individual developers develop code in their free time [12] (hence score = +). For the proprietary software platform we evaluate this as neutral (score = 0): Although this platform owner has to develop the platform on its own, the costs can be amortized over multiple products from different handset makers.

B: Control over Quality. The overall product quality depends on the combination of software from the different contributors and failures often occur because of component interaction, unclearly documented Application Programming Interfaces (API) or unknown use of the system [18, 19]. A firm that controls a wide scope of the architecture can guarantee the product quality more easily. In the situation where multiple firms are involved, and especially when the interfaces are not clearly defined, the quality can easily break down and externally developed code could access data in the system, causing malfunctioning or security problems. Furthermore, applications developed by complementors may not follow the UI standards, as set by the platform owner, thereby causing a reduced user experience [20].

The quality can be controlled easier by the vertically integrated platform type as they have full control over the end-product. Furthermore these firms are able to test the externally developed applications on their handsets (hence score = +). For the proprietary platforms we evaluate this as neutral (score = 0): although they have no control over the hardware, they can control the API and, because of its closed nature, can easier avoid that code with defects is added to the platform. For the open source platform we also regard this as neutral (score = 0). Because of its open nature, faulty code can be added or code can be changed incorrectly by the complementors and applications may compromise the security. An advantage however, is that the software is tested by a variety of firms and open source developers.

C: Flexibility and Variability: Customers increasingly want the software and the handset that serves their specific needs. For instance users have different requirements for the product, e.g. the size of the screen, a keyboard or prefer a low cost device. Specialized firms can often add this variability to the platform more easily [21], e.g. because they have the required knowledge, can reuse existing hardware and software components and they can have a more intimate contact to the end-users [3].

Flexibility and variability is easier to achieve through the open source platform type, since complementors can add or remove functionality without the need to involve the platform owner (hence score = +). For the proprietary platform type we evaluate this as neutral (score = 0): The handset makers can create differentiating products with different hardware configuration; however this is limited to that which is supported by the software platform. As a comparison look at the PC industry where the different OEM suppliers of a Windows based PC can only compete on price, service and hardware quality, since the functionality is large determined by the proprietary software platform. For the vertically integrated platform owner it is far more difficult to cover a wide range of products. In order for a firm to be flexible and respond quickly, it needs to focus on a number of core activities [13] and the development of all the required hardware and software components would simply be too costly. We therefore evaluate that this as challenging for this type (score = -).

D: Freedom to innovate: The optimal definition of the boundaries depends on where the major innovation steps in the architecture take place. When innovation takes place across the boundaries of the platform the integrity of the platform is compromised and the complementors need to be involved thus slowing down the speed of innovation [2]. Therefore, a wide scope allows for larger innovation steps more easily.

The introduction of multi touch screens is such an example. Due to this innovation specialized hardware was needed; the interface towards the user has changed and a new API towards the application developers was required. Such a large innovation step couldn't be done through small changes to an existing platform.

Large innovation steps are easier to establish by the vertically integrated platform type (score = +) because these firms control the entire architecture and complementors do not need to be involved. In the proprietary platform type the innovation is restricted because hardware is not part of the scope and the platform supplier has to involve hardware suppliers and handset makers for major steps (hence score = -). For the open source platform the hardware is also not part of the scope, however, the complementors may change the code and thus has the possibility for innovations, independently from the platform owner, for which the architecture does not have to be changed. Therefore we evaluate this as neutral (score = 0).

E: Efficient use of system resources and hard real time requirements: Due to the need for optimal resource utilization and low power consumption a direct control of the hardware is required. Furthermore, embedded devices have hard real time requirements, e.g. for audio playback and telephone conversations. An embedded device usually contains a System on Chip (SoC) and each component is controlled separately. For instance, for audio playback a separate building block of the SoC is used which can operate with low power consumption and is only active when needed. Furthermore, by controlling each part of the IC separately it can be avoided that

processes interrupt each other. Therefore the design the hardware and software are developed in parallel and require close co-operation [6].

As a comparison, look at the Wintel framework; the dominant ecosystem of Windows and Intel in the PC industry [11]. This is a modular architecture with stable interfaces between the hardware and the software layer. Both Microsoft and Intel can independently innovate on their part of the architecture. Such a modular interface is possible because most demands of the end user can easily be met by existing hardware. For mobile phones such a modular interface is not yet possible since there are still large innovation steps that involve changes to hardware and software together and a modular architecture would lead to less efficient resource utilization [22].

Since the vertically integrated platform type has both control of the hardware and software this can controlled more easily (score = +). In the open source type the complementors can also change the code to accommodate the hardware and vice-versa and therefore this can also be controlled (hence score = +). For a proprietary platform type this is particularly challenging since changes to hardware may require changes to the proprietary platform and vice versa (hence score = -).

F: Stability of the Interface: The success of a platform relies on attracting 3rd party application developers [20]. It is important for a platform to maintain a (sufficiently) stable API, thus avoiding that interoperability problems exist where applications do not work on the variety of devices based on different versions of a platform. This fragmentation is seen as the major challenge by the application developers [23].

Vertically integrated platforms can more easily avoid fragmentation since they have full control over the API (hence score = +). This also holds for the closed source proprietary platform, similarly as the case in the PC industry, which has proven to be the major advantages and success of this type of platform [21] (hence score = +). For an open source platform this is challenging since fragmentation can easily occur because handset makers can change the API or the hardware (hence score = -).

G: Effort for system integration: The time needed to integrate the hardware and software components is taking an increasing amount of time [24]. This is especially the case when components from different parties are used, e.g. with different technologies, non-matching interfaces or heterogeneous architectures [15]. When a modular architecture exists with stable interfaces, the integration time would be substantially less. Furthermore, when multiple parties are involved, the time for interaction, definition of requirements etc., is forming a substantial part of the development effort.

For the vertically integrated platform type the integration is easier to manage since no complementors are involved and the integration is done with in-house development for which the interfaces can be defined (hence score = +). For the proprietary platform type this is particularly challenging since this may require that code has to be altered or glue components have to be developed [25] (hence score = -). In the open source platform type the handset makers or the system integrators can perform the hardware/software integration without (intensive) support of the platform supplier. Therefore we evaluate this as neutral (score = 0).

4.3 Overview of Ecosystems and Their Challenges

The overview in Table 1 shows that for the vertically integrated platform type most factors are easier to address. However, variability is more difficult to achieve and that it has to bear all the development costs.

Table 1. Overview of types of ecosystems and their challenges

Type / Challenge	A: Costs	B: Quality	C: Variability	D: Innovation	E: System resources	F: Interface Stability	G: Integration effort
1:Vertically Integrated	-	+	-	+	+	+	+
2:Proprietary Software	0	0	0	-	-	+	-
3:Open Source Software	+	0	+	0	+	-	0
In this overview: + means: easier to address, 0 means: neutral, - means: particularly challenging							

In the open source software type the complementors can contribute to the platform thus offering better possibilities for flexibility and variability. Furthermore, this type gives the handset makers and system integrators the possibility to tailor the software for specific hardware thus being able to optimize on the system resources. However, fragmentation of the platform can occur more easily.

In the proprietary software platform it is particularly challenging to create innovative products and products with optimal system resources. The latter is a major challenge since a smartphone is an embedded device and a consumer electronics product.

5 Case Studies

This section contains case studies that cover the three types of ecosystems and show how each company addresses the challenges that are described in the previous section. These case studies are based on publicly available information.

5.1 Vertically Integrated Hardware/Software Platforms

Nokia was a vertically integrated firm with the largest market share until 2008. The products were renowned for its quality and long battery lifetime. Nokia offered a large variety of feature and smartphones.

To share the development effort with other handset makers, Nokia's adopted and supported the Symbian platform. The Symbian platform was initially a closed-source collaborative platform of a small group of players and later turned into an open source platform. However, in this ecosystem there was not a strong platform owner, but a divided leadership [8]. This led to a fragmented platform with different branches and

poorly documented APIs and therefore many 3rd party developers abandoned the platform [9]. Although Nokia, being a vertically integrated player, had the possibilities to innovate, they insufficiently took advantage of this because the Symbian platform required too much development effort and many contributors of the platform later opted for Android [8].

The market share of Nokia dropped rapidly since 2008. In 2010 an alliance was created with Microsoft, but no large market share was regained.

RIM entered the market in 1999 with the BlackBerry, one of the first smartphones. Being vertically integrated they could create a differentiating and innovative product, by combing functionality, implemented both though hardware and software, from a PDA, pager and mobile phone. This product was attracted by the professional market because of its functionality and services.

In the years that followed this differentiating functionality became part of products of the competitors: the touch screens took over the QWERTY keyboards and WhatsApp created an alternative for RIMs messaging service. RIM continued to offer a wide variety of handsets for which the development costs could not be amortized with a small market share which has reduced gradually over the last few years.

Apple entered the market in 2007 when hardware components in the mobile domain become commodities, e.g. the first two types of the IPhone used a SoC that was developed by Samsung. The product immediately attracted many customers. Since Apple was already leading in MP3 players and had an online store they could offer the users functionality that went beyond the smartphones that were available.

A strong focus is on better performance and power consumption; especially the latter was a major concern for the earlier handsets. Since they had little experience with developing SoCs and, rather than using a complementor, acquired SoC design firms [26] and other hardware design firms that develop new technologies [27].

Apple uses the advantages of being a vertically integrated player, by focusing on usability and quality, which was considered better than that of the competitors, especially in the early years, thus creating a loyal group of buyers [28]. Furthermore Apple has a strict control over the applications that are offered through their app store by validating each application, prior to making them available in the app store.

Apple avoids development costs and fragmentation by offering only a small amount of variants, by using the same platform for the IPhone, IPod Touch and the IPad, and by restricting the backwards compatibility to two generations of products. Because Apple only develops a small range of variants, their market share is under pressure as many consumers that are attracted to a product with different features, such as a large screen, or a lower cost handset. However, Apple's does not aim for a large market share, but rather focuses on a small set of products for which they can obtain high profit margins [30].

5.2 Proprietary, Closed Source Software Platforms

Microsoft became active in hand held devices in the early 2000s through pocket-PCs and PDA's . At that time there were no other firms that offered a software platform for 3rd parties so the handset makers were willing to pay the license fees.

With each new version of a platform additional hardware is supported, e.g. different screen resolutions and new application processors. Here we see a contrast with the

open source approaches where most additions are done by the complementors. To reduce the integration effort Microsoft started a close cooperation with a small number of hardware platform suppliers and integrators. In 2009 Microsoft closely cooperated with HTC and 80% of the Windows Mobile phones were based on a platform of HTC [30]. In 2010 a close alliance was formed with Nokia to create the hardware platform and handsets. This combination allows for a close co-operation between the hard- and software development that, as shown in the previous section, is required for embedded products and for the speed of innovation. In 2013 Windows acquired Nokia.

Microsoft has created one platform for tablets and PCs and many components are shared with smartphones so that applications can be ported to smartphones as well. Given their dominant position in the PC industry this gives a competitive advantage, but at the same time causes risks because the speed on innovation for smartphones and tablets occurs at a much higher pace then for PCs, for which a stable API is preferred rather than a high degree of innovation.

5.3 Open Source Software Platforms

Google with Android provides a spanning layer that consists of an open source application framework and base applications and uses the open source Linux operating system kernel. In this way Google entered the market with little investments but, since it offers a spanning layer, could become a platform leader. In Android some proprietary Apps are included, such as searching and YouTube, to ensure revenues.

Linux is a real time operating system that is widely used in embedded systems. It offers the developers the possibility to optimize on system resources [31] and supports hard real time requirements [12]. The handset makers were quickly interested in the platform since it offered an alternative for a proprietary platform for which license fees had to be paid [32] and the functionality that Google offered, e.g. Google Maps, attracted many users. Because of the separation of the middleware layer the OS Kernel, the handset makers can make changes to the source code, e.g. to support different screen size, cameras and other ICs. Consequently a large variety of handsets could be developed, resulting in a wide range with different functionality and price settings.

Although Android is an open source system, the main contributions to the source code are done by Google. This gives Google more freedom to innovate because they do not aim for wide consensus. New versions of the platform are offered frequently without focusing on backwards compatibility.

To increase the innovation, Google has created an alliance with Samsung [33]. Samsung develops a wide range of consumer products and develops and manufactures its own hardware. Samsung was able to control the hardware architecture and create innovative products for different market segments. Therefore the speed of innovation and quality of many Android products was possible because an existing large, vertically integrated consumer electronics firm acted as complementor.

In 2011 Google acquired Motorola [34], which was operated by Google as an independent company, to have a complementor that can bring Google's innovation to the market. However in 2014, when already having a large market share was obtained, Motorola was sold again [35]. Only for first-of-a-kind products, such as Google Glass, both the hardware and software are developed by Google.

Fragmentation of the Android platform is seen as a major concern by the developers [23]. As an example, look at the game "Angry Birds" that was initially introduced for the iPhone and following its success developed for other platforms. When developing a version for Android the developers ran into performance problems of different devices. As a result the game was not available on a large group of devices, i.e. devices with different hardware or based on different versions [36]. Google is taking measures to avoid fragmentation by defining a set of baseline compatibility standards. To have a better control over the quality, in recent versions Android also incorporates stricter security control [37] and provides a compatibility test suite [38]. Furthermore Google is making agreements with handset makers to use the standard UI and set of Googles Apps, which are more consistent of that of the complementors, by adding tighter conditions for the use of the App store by the handsets [39,35].

Meamo/Tizen, Firefox OS, Ubuntu OS, Sailfish OS Over the last few years several other attempts have been made to create an open source platform ecosystem to replicate the success of Android, all of which are based on Linux. An example is Tizen which was born out of MeeGo, which was itself was combination of Moblin and Maemo. None of these initiatives created an attractive alternative because there was no firm taking a role as orchestrator. Firefox OS, also based on Linux, aims for a better support of the increasing amount of HTML5 websites through a more direct control of the hardware [40]. However, Firefox OS also offers an API for native Apps since HTML5 doesn't support the required functionality, especially for games.

In 2013 Samsung abandoned Bada, their proprietary operating system, to contribute to Tizen [41]. When Samsung would replace Android by Tizen for all their products, this could shift the balance for this type of ecosystem.

5.4 Summary of the Case Studies and the Mitigation Strategies

The table below shows how the key players, i.e. the platform leader with the largest market share of each ecosystem type, mitigate the challenges as described in Table 1.

Table 2. Mitigation actions on the main challenges

Ecosystem type and main challenges	**Mitigation actions of the key player**
(1) Vertically Integrated: Costs & Variability	Apple limits the variability and development costs by focusing on a particular market segments and it only supports the latest versions of the handsets.
(2) Proprietary Software: Innovation, System resources & Integration effort	Microsoft closely cooperates with a small group of HW platform suppliers and recently acquired Nokia, a previously vertically integrated firm.
(3) Open Source Software: Interface stability & Fragmentation	A compatibility test suite is offered with baseline compatibility standards. Agreements are made with the handset makers to use standard Google Apps and User Interface.

6 Historical Perspective and Scenarios for the Future

The table below shows the market share of the three types of platforms. This shows that the vertically integrated platforms initially had the largest market share, but in later years the open source platform.

Table 3. Market share of different ecosystem types for smartphones

	2007	2008	2009	2010	2011	2012	2013
Vertically integrated hardware/software platforms	87 %	88 %	87 %	73 %	51 %	31 %	18 %
Proprietary, Closed Source Software Platforms	13 %	12 %	9 %	4 %	2 %	3 %	3 %
Open source software platform	0 %	0 %	4 %	23 %	47 %	66 %	79 %
Total sales in millions of units	122	139	172	298	471	675	967

(Note: This overview is for smartphones only, since for tablets insufficient data was available. This overview is compiled from reports from Gartner [42]).

This change in market share can be explained because in the early years the vertically integrated firms were able to innovate faster and the end-users were willing to pay for new functionality. When the innovation slowed down, the open source software platforms gained market share because handsets were offered with more variability and with a lower price.

When the degree of innovation further slows down, the computing requirements may meet the user's demands, allowing for a modular architecture to be created that provides a clear separation between the software and hardware layer [43]. The advantage of such a modular architecture is that integration will become easier which will especially benefit the proprietary software platform types. Therefore this type of ecosystem might gain market share, as was the case with the PC industry in the 80s [11]. An important uncertainty is whether tablets will replace PCs in a business environment. When this happens, the main supplier, i.e. Microsoft, has a strategic advantage since they are dominant in the PC market.

When PCs and tablets remain consumer devices and the speed of innovation does not allow for a more modular architecture, the types of ecosystems that are now dominant are likely to keep their dominant position.

7 Comparison with Related Art

In our previous work [1] we analyzed the transition of mobile devices from 1990 until 2010 and described a model for different industry structures. The current paper analyses the situation of today. Some of the forces that have caused the transition are used in the current paper as factors to compare the three types of ecosystems.

Another previous work of the co-author [20] describes architectural challenges. Some of these challenges have been used in the current paper. However in that work the challenges have not been identified for mobile devices or embedded systems.

The work of Gawer and Cusumano [4, 12] discusses the scope and technology design of a platform but does not discuss architectural implications nor does this work identify or compare different types of ecosystems for mobile devices.

Other related work identified different types of ecosystems [3, 16], but this work did not compare ecosystems in a particular domain nor did it identify factors that determine the challenges from architectural perspective.

Several predictions on market shares have been done by industry analysts, e.g. by the Gartner group [42], but none of these used architectural aspects as a basis.

8 Conclusions and Further Research

In this paper we identified three different types of ecosystems for smartphones and tablets and identified the factors that determine the optimal scope and technology design. Based on this analysis we described the challenges for each type of ecosystem from an architectural point of view.

The case studies demonstrated how different firms act on the factors identified in this paper: Apple takes over existing hardware design firms and focusses on a limited variety of products. Microsoft, who did not have a hardware department nor created an end product, took over an existing hardware firm to be able to act as a vertically integrated HW/SW ecosystem type. Google, who is faced with a high degree of fragmentation, is offering a compatibility test suite and makes agreements with the handset makers to use standard Google Apps and User interface.

Our analysis shows that when the speed of innovation slowed down, most of the vertically integrated firms were not able to amortize the development investments and an open source platform obtained the largest market share, which, in comparison with the closed proprietary platform, allows more flexibility and variability and efficient use of the system resources. In this paper we predicted that when the market requires a more stable interface, e.g. when smartphones and tablets need to support business applications, the proprietary platforms can obtain a larger market share.

We think that the types of platforms that are identified in this paper and the factors that determine the scope, is applicable to the wider domain of consumer electronics and embedded systems, but further research is necessary.

The analysis in this paper is based on information in literature and the experience of the authors. Further research, e.g. with multiple experts, may give a more precise comparison and may reveal more relevant factors.

The results of this paper can be used by the platform owners to reconsider the scope of the platform, e.g. by expanding the scope to areas where more control over the quality is preferred, or to publish part of the platform as open source so that handset makers can add variability more easily. On this topic further research is needed.

The analysis used in this paper can provide to the academic community a lens to evaluate the strategies in the market of mobile devices from an ecosystem and architectural perspective.

References

1. Hartmann, H., Trew, T., Bosch, J.: The changing industry structure of software development for consumer electronics and its consequences for software architectures. The Journal of Systems & Software 85, 178–192 (2012)
2. Messerschmitt, D.G., Szyperski, C.: Software Ecosystem. MIT Press (2004)
3. Bosch, J.: From Software Product Lines to Software Ecosystems. In: SPLC. Sheridan (2009)
4. Gawer, A., Cusumano, M.: Platform leadership. Harvard Business School (2002)
5. Jobs, S.: Thought on Flash (April 2010), http://www.apple.com/hotnews/thoughts-on-flash/ (accessed November 24, 2011)
6. Halasz, M.: Menu expands at the OS diner. Timesys Corporation (February 14, 2011)
7. http://www.qualcomm.com/solutions/operating-systems (accessed January 2014)
8. West, J., Wood, D.: Tradeoffs of Open Innovation Platform Leadership: The Rise and Fall of Symbian Ltd., Social Science and Technology Seminar Series (2010-2011)
9. Gilson, D.: The History of Symbian's Secret Fragmentation (March 12, 2012), http://www.allaboutsymbian.com/
10. Baldwin, C.Y., Clark, K.B.: Managing in the Age of Modularity. Harvard Business Review, 81–93 (September/October 1997)
11. Grove, A.S.: Only the Paranoid Survive: How to exploit the crisis points that challenge every company and career. Currency Doubleday (October 1996)
12. Gawer, A., Cusumano, M.: How companies become platform leaders. MIT Sloan Management Review (2008)
13. Bolwijn, P.T., Kumpe, T.: Manufacturing in the 1990s—Productivity, flexibility and innovation. Long Range Planning 23(4), 44–57 (1990) ISSN 0024-6301
14. Sheate, W.R.: Tools, Techniques and Approaches for Sustainability. World Scientific Publishing Company (September 30, 2010)
15. Henzinger, T., Sifakis, J.: The embedded systems design challenge. In: Proceedings of the 14th International Symposium on Formal Methods, FM (August 2006)
16. Popp, K., Meyer, R.: Profit from Software Ecosystems. Books on Demand (2010)
17. van Genuchten, M.: The Impact of Software Growth on the Electronics Industry. IEEE Computer (2007)
18. Trew, T., Soepenberg, G.: Identifying Technical Risks in Third-Party Software for Embedded Products. In: Fifth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (2006)
19. Oberhauser, R., Schmidt, R.: Improving the Integration of the Software Supply Chain via the Semantic Web. In: International Conference on Software Engineering Advances (2007)
20. Bosch, J.: Architecture in the age of compositionality. In: Babar, M.A., Gorton, I. (eds.) ECSA 2010. LNCS, vol. 6285, pp. 1–4. Springer, Heidelberg (2010)
21. Moore, J.F.: Businesses ecosystems and the view from the firm. The Antitrust Bulletin (March 2006); American Antitrust Institute
22. Christensen, C., Verlinden, M., Westerman, G.: Disruption, disintegration and the dissipation of differentiability. Industrial and Corporate Change 11, 955–993 (2002)

23. Gadhavi, B., Shah, K.: Analysis of the emerging android market, San Jose State (2010)
24. Underseth, M.: Verifying Embedded Software Supply Chains. EETimes (April 2007)
25. Hartmann, H., Keren, M., Matsinger, A., Rubin, J., Trew, T., Yatzkar-Haham, T.: Using MDA for integration of heterogeneous components in software supply chains. Science of Computer Programming 78(12), 2313–2330 (2013)
26. http://www.eetimes.com/document.asp?doc_id=1168401 (retrieved December 24, 2013)
27. http://www.reuters.com/article/2013/11/25/us-primesense-offer-apple-dUSBRE9AO04C20131125 (accessed January 2014)
28. Jones, C.: Apple Vs. Samsung: Who Could Win The Smartphone War? Forbes (August 20, 2013)
29. Grobart, S.: Apple Chiefs Discuss Strategy, Market Share and the New iPhones (September 19, 2013), http://www.businessweek.com
30. Tricia Duryee: We Learned Just How Great Of A Partner HTC Is To Microsoft. (February 17, 2009), http://paidcontent.org/ (accessed January 2014)
31. Weinburg, W.: Time and Space: Optimizing Boot Up and Footprint in Linux –Based CE Devices. Open Source Development Lab (2006)
32. Hansel, S.: Microsoft, Google and the Bear. New York Times (October 26, 2009)
33. http://www.samsung.com/us/news/3475 (accessed January 2014)
34. http://www.google.com/press/motorola/ (accessed January 2014)
35. http://www.uswitch.com/mobiles/news/2014/01/samsung_and_google_thrash_out_deal_over_android_bloatware/ (accessed January 2014)
36. http://www.techhive.com/article/211152/angry_birds_devs_angry_at_android_fragmentation.html (accessed January 2014)
37. Sierraware: “Android: Is It Secure Enough?”, http://www.sierraware.com/SE_Android_Security_Integrity_Management.pdf
38. Android compatibility test suite, http://source.android.com/compatibility/cts-intro.html (accessed January 2014)
39. Gannes, L., Fried, I.: After Google Pressure, Samsung Will Dial Back Android Tweaks, Homegrown Apps. (January 29, 2014), http://recode.net/2014/01/29
40. http://www.mozilla.org/en-US/firefox/os/
41. Byford, S.: Samsung finally folding Bada OS into Tizen (February 25, 2013), http://www.theverge.com/
42. Gartner reports on smartphone market shares 2007 – 2013), e.g., http://www.gartner.com/newsroom/id/2623415 (accessed January 2014)
43. Christensen, C., Anthony, S., Roth, E.: Seeing what’s next. Harvard Business School (2004)

ESAO: A Holistic Ecosystem-Driven Analysis Model

Jan Bosch[1] and Petra Bosch-Sijtsema[2]

[1] Chalmers University of Technology, Department of Computer Science and Engineering, Software Engineering, Gothenburg, Sweden
[2] Chalmers University of Technology, Department of Civil and Environmental Engineering, Construction Management, Gothenburg, Sweden
{Jan@JanBosch.com, Petra.Bosch}@chalmers.se

Abstract. The growing importance of software ecosystems and open innovation requires that companies become more intentional about aligning their internal strategy, architecture and organizing efforts with the ecosystem that the company is part of. Few models exist that facilitate analysis and improvement of this alignment. In this paper, we present the ESAO model and describe its six main components. Organizations and researchers can use the model to analyze the alignment between the different parts of their business, technologies and ways of working, internally and in the ecosystem. The model is illustrated and validated through the use of three case studies.

Keywords: Ecosystem, Strategy, Architecture, Organizing, Model.

1 Introduction

Recently, more and more research discusses the relevance of ecosystems for companies as well as market segments. This research lifts up the business aspects of ecosystems, the innovation element as well as the more technical component of ecosystems. An ecosystem is defined as an economic community supported by a foundation of interacting organizations and individuals, which can also be perceived as the organisms of the business world [1]. The terminology of business ecosystem defines ecosystems as consisting of three characteristics [1, 2, 3]: (a) a symbiosis relationship in which the survival of all members implies the survival of the ecosystem. (b) Co-evolution in which partners co-evolve capabilities around new innovations and finally (c) ecosystems are often based on a particular platform, which is defined as tools, services or technologies used in the ecosystem that enhance performance of its members [4, 5, 6]. Especially in the software industry the term software ecosystem has gained enormous popularity and can be defined as: 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 [7, 8].

Current ecosystem research primarily looks at the ecosystem – but does not link this back towards the internal organization or to the implications of the internal organization, software platform or architecture and the ways of organizing and

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 179–193, 2014.

working. This is a challenge for many organizations for several reasons. First, the way the organization works internally and the way the company engages with its ecosystem need to be closely aligned with each other, as a strong co-dependency exists between the two. Second, as companies increasingly seek to focus their internal efforts on what truly differentiates them and try to outsource as much as possible of the non-differentiating activities, the reliance on the ecosystem is increasing significantly. Finally, as customers are transitioning from buying products to acquiring services, the burden of integrating different products into a dedicated solution for customers is falling to ecosystem players that need intimate interactions with other players in the ecosystem. However, few, if any, holistic models exist that support organizations to gain a better insight for aligning the complex internal and ecosystem dimensions of R&D.

In this article we propose a model that encompasses elements of both the ecosystem as well as the internal organization. The model provides three perspectives, i.e. strategy, architecture and organizing, and two dimensions, i.e. internal and ecosystem. This provides an approach where the alignment between the perspectives and between the internal and ecosystem dimensions can be analyzed and improved.

The remainder of this paper is organized as follows. Below we first describe the problem statement. In section 3, we introduce the ESAO model and describe its six main components. Section 4 is concerned with validating the ESAO model with three case study examples where we illustrate the alignment, or lack thereof, between the six components of the ESAO model. Finally, we conclude the paper and discuss the core contribution and its implications.

2 Problem Statement

With the growing proliferation of open innovation and software ecosystems, it becomes more and more important that software research takes a more holistic picture instead of describing individual elements. The vast majority of research studies one dimension of a software intensive organization. These different dimensions are, for example, the software ecosystem, the architecture, as well as the organizational aspects of software intensive organizations. The software ecosystem literature primarily studies the ecosystem and discusses different roles in the ecosystem as well as a strong focus on the keystone player in a particular ecosystem [8]. In the software architecture literature there is a strong focus on the design and evolution of the architecture as well as on the technology choices required to support software architecture [9]. In the agile community there is a lot of focus on how to organize agile teams as well as processes and ways of working [10, 11, 12]. However, virtually all research takes a narrow and deep approach, focusing on a specific aspect. In our review of the literature, we found very few models that provide a holistic and complete overview of all these different dimensions and their interdependency.

One often applied model that provides a holistic perspective of the end-to-end dimensions of business, technology and organization is the BAPO model [9, 13]. The BAPO model defines for four independent software development concerns: (1)

Business, concerned with how to make a profit, (2) Architecture, concerned with the structure of and technologies required to build and evolve the software system, (3) Process, defining the roles, responsibilities and relationships within software development as well as the tooling and ways of working and, finally, (4) Organization, defining the actual mapping of roles and responsibilities to organizational structures [13]. The model is frequently applied for analysis and assessment in both academia and industry. In the figure below, the BAPO model, as well as the intended dependencies in the model, are shown.

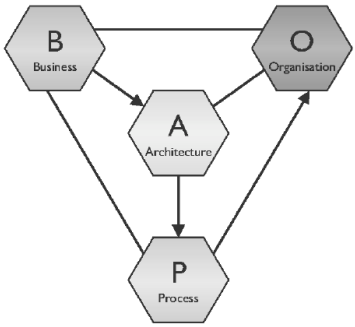

Fig. 1. The BAPO concerns (source: [13]).

One of the authors of this paper was involved in the development of the original BAPO model. However, in ten years of evolving an understanding of the problem, we can identify a number of challenges with the original BAPO model. These challenges are the following:

- BAPO is a model that only incorporates the internal organization, but does not take into account the external environment like the ecosystem. From software ecosystem literature we know that the ecosystem has a major impact on an organization [7] and therefore it is important that an ecosystem dimension is part of such a model.
- The BAPO model was originally developed in the context of software product line research and in its detailed definition assumes domain software and product software. This limits the applicability of the model in practice as not all companies are using software product lines. Although not impossible, the model is less applicable to companies that are less reuse-centric.
- BAPO strongly enforces a B➔A➔P➔O sequence. In practice, however, this represents too much of a simplification of the reality in the organizations that we engage with. Even though idealistically speaking the sequence of the B (business) should drive the A (architecture), A should drive the P (processes) and P should drive the O (organization structure), in practice one never starts from a green field situation. Consequently, one has to allow for bi-directional dependencies and the focus should be on achieving alignment between the four dimensions.

Having analyzed these challenges, we came to the conclusion that there is a need for a new improved model that allows organizations to accomplish the following:

- Serve as a holistic analysis framework
- Serve as a benchmark for effective software product engineering
- Support the assessments of software product engineering for capability
- Evaluate software production units, divisions, or companies
- Support the improvement of software product engineering, which involves producing assessments and improvements plans

The list above is a quite extensive list of requirements on any model. In this paper, however, we focus specifically on the analysis model, i.e. the first item, and its facilitation of alignment between the different aspects of a software-intensive systems organization.

3 The ESAO Model

Based on the discussion above, we propose an extension and evolution of the BAPO model, i.e., the ESAO model. The model is based on our experience from working with dozens of companies around R&D management topics. As we have outlined in the earlier parts of the paper, we have increasingly identified challenges with the tools that we had available to help companies and in response, we have developed a new model. The ESAO (Ecosystem, Strategy, Architecture and Organizing) model consists of six interdependent and interconnected dimensions that are important to take into account for software development. The six dimensions of the 'ESAO' model concern both an internal company and an external company perspective.

In the remainder of this section, we first describe the internal perspective and subsequently we discuss the ecosystem perspective.

3.1 Internal Perspective (SAO)

The internal perspective consists of three main dimensions, i.e. strategy, architecture and organizing. Below, each of these dimensions is defined in more detail.

1. ***Internal Company Strategy***: The strategy of the company lays down the basis for the future path of the firm concerning the business. In particular, the strategy is concerned with how the company generates revenue now and in the future. The company strategy is relevant for the internal prioritizations and decisions made within an organization, and is closely related to the software development strategy and architecture. The internal business model development is part of the internal strategy. The business model defines how the firm creates and delivers value to customers and then converts payments received to profits [14]. The internal company strategy can be related to the Business concern of the BAPO model.

2. ***Internal Architecture***: The architecture comprises the technical structure means to build the software-intensive system as well as the technology choices. The company strategy defines which aspects of the business will develop going forward and need to be prioritized and which can be deprioritized. This is important input for the architecture decisions as it allows effective management of future evolution cost. The internal architecture dimension can be related to the Architecture dimension of the BAPO model.
3. ***Internal Organizing***: The ways of organizing work, way of working, roles, responsibilities, processes and tools within software development are important and closely related to the architecture and strategy of the firm. In an earlier publication one of the co-authors developed the concept of 'Stairway to Heaven', to describe how development typically evolves over time [16]. This element is related to the Process and partly Organization part of the BAPO model. The ESAO model combines the P and O parts of the BAPO model as, in practice, the adoption of agile approaches assumes empowered, cross-functional teams and the locus of power is much more with the teams than with the traditional reporting hierarchies. As a consequence, the precise organization structure is less important than earlier and the focus has shifted to organizing the work.

3.2 External Ecosystem Perspective (ESAO)

In the ESAO model, we use the same three dimensions discussed above for the external ecosystem. However, depending on the role of the organization in the ecosystem, the company has more or less power in its ecosystem. When discussing the ecosystem as an important dimension in relation to the internal strategy, software architecture and way of organizing, it becomes relevant to understand that firms can obtain different roles within an ecosystem. These roles are often discussed and defined in literature and are also lifted up in the next section concerning validation of the model with help of three case studies. The main roles often studied in ecosystems are the following:

- Central firm or also called the keystone or platform firm who is the dominant player and orchestrator in the ecosystem [2, 3, 4, 5, 6, 8].
- Complementors and component players who provide a product or service that complements the platform or product of an ecosystem and enhances the value of the platform [5, 8].
- Integrators who brings together the parts provided by different ecosystem players into an integrated solution for the end-user. Depending on the ecosystem, this role can be played by the keystone player, the end-user or a separate organization [5, 8].
- A final role important in the software ecosystem is the end-user [8].

The role that the organization plays in its ecosystem determines the amount of freedom that it has in terms of defining its strategy, architecture and organizing

dimensions. However, even complementors should not view themselves as powerless. Instead, every player has a set of strategic options available to optimize their position and future outcomes:

1. ***Ecosystem Strategy:*** The external strategy of a company is related to the business and software ecosystem of the firm and the strategic options that it has available in its current role in the ecosystem. As a keystone player strategic decisions are concerned with providing a viable business model for complementors while maximizing its own revenue. In addition, whether the complementors should be encouraged to compete or if the focus should be on collaboration (cf. [15]). For complementors, the goal often is to maximize its own stake in the ecosystem. One strategy is to seek to form a niche market in the ecosystem, to become the keystone partner in that niche and to expand from that position of strength. For integrators, the relationship to the end-user and maximizing its own visibility while diminishing the role of other ecosystem players is often a viable strategy to increase its relevance. Depending on the strategic choices made by the company, there are significant implications on the system and software development of the firm.
2. ***Ecosystem Architecture***: The ecosystem architecture defines the interface between the internal architecture and the solutions that are provided by ecosystem partners in terms of the following:
    a. The interface between my firms suppliers and my firm
    b. The interface between firms that build software on top of my product or platform. These roles are also discussed as complementor roles [5, 8] and they can deliver, add, or develop components and complements to a product or platform as a complement to your firm's platform or product.
    c. The interface between my firm and firms that operate in the same ecosystem role as my firm, but that provide other types of functionality.
    d. Finally, depending on the player providing the integration of ecosystem solutions, the interface between my firms and integrators.

    In addition to the focus on interfaces, the focus is also on the architecture strategy. As we discussed in [17], there is a constant commoditization process ongoing that requires that ecosystem players pro-actively innovate around new functionality and release commoditizing functionality to other players or the open-source community.
3. ***Ecosystem Organizing:*** Deals with how firms work with their customers, suppliers, and ecosystem partners in terms of processes, tools used, ways of working, and ways of organizing the collaboration. For instance, in some of the companies that we work with, the company has internally adopted agile ways for working and continuous integration. However, the suppliers of the company still use traditional waterfall or iterative development causing the supplied parts of the system developed by the company to be updated very

infrequently. This causes significant disruptions in the internal development processes.

In the figure below, we present the ESAO model graphically (figure 1). We aim to illustrate two important aspects. First, in order to optimize synergy, alignments between the internal and the ecosystem dimension, the strategy, architecture and organization perspectives become important. For instance, a business strategy that stresses high responsiveness to customer requests does not combine well with a traditional, iterative development approach with new releases of software every six months.

Second, equally important, the internal and external perspective on each dimension should be in line with each other, i.e. ecosystem strategy and internal strategy, ecosystem architecture and internal architecture as well as ecosystem organizing and internal organizing. For instance, the aforementioned example of the company using agile methods internally and waterfall methods with its suppliers illustrates the inefficiencies caused by lack of alignment.

A change in one of the dimensions has consequences for the other dimensions. Although no organization will every be completely aligned in all six dimensions of the ESAO model, our experience has shown that there is significant benefit to continuously seek alignment when implementing changes in one of the six dimensions. That allows misalignment to be short-lived and affecting the organization as little as possible.

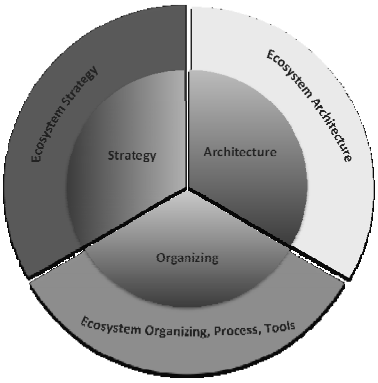

Fig. 2. The ESAO model with internal and ecosystem dimensions.

The ESAO model does not insist on a particular order, but instead focuses on achieving and maintaining alignment between the different dimensions. The model could be related to the research methodology of systems thinking which encompasses a holistic approach in which often an external and internal focus is taken into account. However, the system that is studied can vary depending on the definition and boundaries set for a system. The ESAO model provides a framework that supports the analysis of both the external and internal firm.

4 Validation

We validate the use of the ESAO model with the help of three individual cases. In all three cases either the ecosystem changed or the position of the firms in their ecosystem changed and this had implications for the internal as well as ecosystem strategy, architecture as well as way of organizing in the different firms. With help of the cases we show that being able to analyze, assess, and react to the external dimension, i.e., the ecosystem dimension is important for firm's strategy, internal development and way of working. Below we use the ESAO model as an analysis framework for three different cases in which we introduce the case, discuss the change trigger and analyze the case according to the ESAO dimensions. Due to limitations in space of this paper, we can only describe the findings and analysis rather briefly. The examples described below are extracted from longer-term data collection through interviews, workshops and discussions between researchers and managers and R&D engineers. For case study Alpha we held 13 interviews in the firm, and had numerous workshops with complementors and end-users (total of 21 people). In case Beta we primarily held 4 group interviews and workshops with 5-10 people (total of 20 people) and in case Zeta we held 14 group interviews with 5-10 people (total of 50 people). Three case studies cannot give sufficient generalization for the model, but give an insight in how the model can be applied to analyze the complexity of software development R&D. In the anonymous cases we discuss below we show that a change in one of the six factors has implications for the other ESAO factors. The interviews were held retrospectively in order to capture the full implications of these external and internal changes. The companies were therefore selected based upon the fact that strategic changes were implemented in their firm.

4.1 Case Alpha: Ecosystem Changes

Introduction to Case Alpha. Company Alpha is a Fortune 1000 company developing software products and services operating, primarily, on personal computers. The company's products address both consumer and business markets and the company releases several products per year, including new releases of existing products and completely new products. The products developed by the company range in the multi- to tens of millions lines of code and tend to contain very complex components that implement national and international regulations. The case concerns one of the products of the company that has a user base of millions. For this product, the company is the keystone or dominant player in their ecosystem.

Change Trigger. The case study is dealing internally with a changing ecosystem. The company had treated its entire customer base as a relatively homogeneous population; however, based on market research and customer feedback it became increasingly clear that many customer segments existed with unique and specific needs. On the other hand with a customer base numbering in millions, there was a growing base of developers of both within case Alpha as well as outside of the firm (i.e. complementors), that in various non-endorsed ways sought to extend the functionality

in the base product with features for individual customers or narrow customer segments. From the interviews it was stated that the company could never serve all these segments in a cost effective manner.

ESAO Analysis

Ecosystem Strategy: the case company has a keystone role within their ecosystem concerning this particular product. For many years the company either ignored or actively discouraged third party developers to extend its product. A number of years ago with the advent of the iPhone apps the company decided to adopt an alternative ecosystem strategy. It decided to copy the Apple app store model and collect 30% of the sales generated by 3rd party developers.

Ecosystem Architecture: the change in ecosystem strategy caused the architects to introduce an ecosystem API to the product. However, as the product managed quite sensitive data for its users, the company introduced a multi-layered API were certified apps would get more access, and non-certified apps only received read-only access.

Ecosystem Organizing: the company decided to pro-actively engage with its developer community through the organization of developer conferences, regular newsletters and other forms of communication. In addition, the company created a certification mechanism that allowed 3rd party developers to certify their application. Finally, the company introduced a market place inside its product that managed payments (inside apps) as well as entitlement for 3rd party developers.

Internal Strategy: the predominant change in business strategy for the company concerned the best ways of serving customer segments. Before the adoption of the ecosystem strategy, the ongoing debate within product management, concerned the introduction of customer segment specific functionality, versus the increased complexity of the product for customers in different segments. After the adoption of the ecosystem strategy, interviewees mentioned that the discussion changed and focused on the boundary between functionality that should be in generic products within the platform and functionality that should be left to the developer community.

Internal Architecture: the impact on the product architecture is two-fold.

(1) The ecosystem API was introduced which required a careful analysis of which parts of the product internals were to be exposed and which would remain hidden.
(2) As the company adopted a certification mechanism, the ecosystem API, as well as the rest of the product, had to support the differentiation between certified and non-certified apps.

Internal Organizing: the primary change in the internal organization was the development of a unit responsible for 3rd party developers. This unit was both responsible for certification of apps, as well as for maximizing adoption of the product platform by 3rd party developers. Internally, this unit became the champion

for 3rd party developers. Additionally the implementation of certification and API also implied new ways of working within the firm.

Reflection: In this case it was clear that the change trigger initiated from the ecosystem organizing level, in which the firm noticed changes in the way of interacting and collaborating with customers and developers. As is clear from this case, when a product reaches a customer base number in the hundreds of thousands and millions, there will be significant pressure by both customers and 3rd party developers, to 'open up the product' for customer and customer segment specific extensions. The patterns that we described in this case, is, we believe, quite generic for companies in this situation.

4.2 Case Beta: Pushed Back in Value Chain

Introduction to Case Beta. The case company Beta is a large global company in the embedded system domain. The unit that we studied works with OEM customers (Original Equipment Manufacturers) to provide one of the major sub-systems in their product. The company worked with the OEM's in the form of a solution provider, and delivered a dedicated subsystem implementation in response to the requirements from the OEM. The revenue of the company was generated by subsystem unit sales, where a subsystem unit consists of mechanical, hardware and software unit parts. Although the company provides software development services for its OEM customers, this was a negligible part of their business and received very little attention from general management.

Change Trigger. Until recently, the case study company Beta provided the complete solution to its OEM customers. Interviewees mentioned that some years ago, a shift started to occur in the unit's ecosystem. First, OEMs were starting to demand that software provided by the OEMs would be needed to be integrated in the overall solution, frequently replacing functionality developed by the case study company Beta. Second, OEM customers began to demand that the case study company provided arbitrary compositions of hardware, software and mechanics provided by competitors, the OEM and the case study company. For instance, in some cases, the case study company was requested to provide its software on hardware developed by competitors. In another case, the competitor software needed to be deployed on hardware developed by the case study company. The most complicated situations, however, were where OEM software, competitor software and software developed by the case study company needed to be integrated. The architectural boundaries in the three software subsystems did not align with each other, requiring deep integration and rework of already developed components to accomplish functional integration while achieving the necessary quality attributes.

During this shift the company's role in their ecosystem shifted from a turnkey solution provider to either a component provider or an integrator that worked under the close supervision of the OEM.

ESAO Analysis

Ecosystem Strategy: the role of the case company shifted from a turnkey solution provider to a component and integrator role. The case had to respond to what was happening in their ecosystem. The company explicitly designed the industry specific standardized architecture to coincide as much as possible with the sub system interfaces already existing in their platform architecture. This strategy would decrease the integration cost of customer and competitor subsystems that needed to be integrated into the company's products.

Ecosystem Architecture: the change in ecosystem strategy was driven by significantly increased integration costs, which severely impacted the profitability of the business. Due to this strategic change, the ecosystem architecture evolved into a much more modular architecture that allowed for replacement of subsystems of components provided by other parties.

Ecosystem Organizing: The case company used the existing standardization body to drive their standardization efforts. The case had to respond to the changes in their ecosystem by changing their role in the ecosystem. Their initial role was being a full solution provider but now the case study has shifted towards an integrator role and complementor role in their ecosystem.

Internal Strategy: the company went from an almost exclusively unit-based business model, towards a business model in which it split its units into three:

(1) To a hardware-mechanical sales unit model that was based on the traditional sales unit model,
(2) A software license sales and
(3) Consulting service business where the organization would build and integrate software for any hardware mechanics configuration that the customer desired.

Internal Architecture: The internal architecture changed towards increased modularity. The original architecture was a highly integrated architecture towards optimizing hardware optimizing efficiency. This architecture was evolved into one into where modularity and decoupling between subsystems was prioritized at the expense of resource efficiency.

Internal Organizing: the changing business model and the increased architecture modularity caused the following changes in the organization:

(1) For each type of business model a separate organizational unit was created. Especially in software this lead to a stronger separation between product platform development and customer projects.
(2) In order to increase responsiveness to customers the interviewees mentioned that they had adopted agile work practices and are currently implementing continuous integration practices.

Reflection: In this case the changes in the ecosystem architecture with demands for integration triggered the other dimensions. As is clear from the description of the six dimensions of the ESAO model, there were several bi-directional interdependencies in case Beta, such as between the ecosystem and internal counterparts, as well as between the three aspects. For instance, although the organization identified that a shift in the ecosystem was taken place, no action was initiated until the integration costs for customer projects became unacceptably high. In order words, the Internal Organization initiated the change in all other dimensions, showing clearly that there is no sequential change process like BAPO, but a continuous alignment of the different dimensions.

4.3 Case Zeta: Strategic forward Integration

Introduction to Case Zeta. The case study company Zeta is a global company in the embedded systems industry. The industry in which case Zeta operates is highly fragmented with no company having more than 5% market share. The company has thousands of competitors, but it has been able to differentiate itself using product quality and reliability as key competencies. The products of the company have traditionally consisted of a major mechanical component and a minor hardware and software component. However, over the last decade, the R&D investment has shifted in a quite significant fashion towards software development. The company operates in a complicated technology ecosystem, consisting of wholesalers, retailers, installers, specification engineers, maintenance and end-customers.

Change Trigger. Over the last decade, many new competitors, especially in Asia (India and China), have appeared on the market, causing significant change in the ecosystem. Until recently, the original differentiator and niche of the company, i.e., quality and reliability, were sufficient to justify its market share and pricing power. During recent years, the quality of competitor products has reached a level that this differentiation strategy was no longer viable. After analyzing its strategic options, the company decided to forward integrate into its ecosystem and to start offering systems and solutions for which it originally only provided some of the components. The company started to change towards a turnkey provider with complex solutions in the HVAC industry (Heat, Ventilation and Air Conditioning). The reason for this is that the margins on systems and solutions were an order of magnitude higher than the margins of its traditional products.

ESAO Analysis

Ecosystem Strategy: as the company decided to forward integrate in its ecosystem, the company changed its role in the ecosystem from a component role towards a solution provider role. One of the main challenges of this change was to avoid upsetting its existing customer base. To accomplish this the company focused its new offerings initially on geographies were its primary customers have little or no market presence.

Ecosystem Architecture: the company originally provided its products as closed systems with minimal ability for system integrators and solution providers to access the product. Its ecosystem architecture strategy concentrated on two factors:

(1) The adoption of industry standards to simplify the integration of its products as well as products from other manufacturers into systems and solutions.
(2) It provided an ecosystem API that allowed third-party developers as well as its internal systems and solutions groups to extend basic product functionality with systems and solutions specific functionality.

Ecosystem Organizing: the company was looking to simplify the integration of products from other manufacturers into its own systems and solutions and had little incentives to market the capabilities to other players in its ecosystem. Consequently, it intentionally limited communication and interaction with others in the ecosystem.

Internal Strategy: although its product margins were under pressure, the company still was able to sell its products at a reasonable margin. Consequently, it adopted a dual business strategy.

(1) One the one hand it sought to maximize the scale in its produce manufacturing and sales, seeking to drive down costs by maximizing scale.
(2) On the other hand, it pro-actively build a number of system and solution business that, though high margin, were only able to provide limited unit sale levels.

Internal Architecture: the product architecture was optimized for minimizing hardware resource cost but allowed for extension with 'apps' through its ecosystem API. The tension in the organization was between minimizing hardware resource cost as desired by the product sales organization, and providing excess hardware resources in order to allow for system and solution specific apps to be installed on the device.

Internal Organizing: although the organization considered to create separate R&D departments for products and systems and solutions R&D, it instead developed a governance mechanism that allowed one R&D department to satisfy the hardware and software needs of the product systems and solutions business units. The governance mechanism consisted of a board representing all relevant stakeholders that met frequently, and prioritized the needs of products, systems and solutions.

Reflection: Based on the complex ecosystem strategy of the firm, and their role in the ecosystem, the company pro-actively decided to change its business strategy rather than being forced by other players in the ecosystem. However, again, this change had affect on all dimensions of the ESAO model. As we sought to highlight in this case, the importance is the alignment between the six dimensions, not which dimension initiates the change.

5 Conclusion

Software ecosystems and open innovation are increasingly important for companies as well as market segments. Current ecosystem research primarily looks at the ecosystem – but does not link this back towards the internal organization or to the implications of the internal organization, software platform or architecture and the ways of working. This is a challenge for many organizations for several reasons. First, the way the organization works internally and the way the company engages with its ecosystem need to be closely aligned with each other, as a strong co-dependency exists between the two. Second, as companies increasingly seek to focus their internal efforts on what truly differentiates them and try to outsource as much as possible of the non-differentiating activities, the reliance on the ecosystem is increasing significantly. Finally, as customers are transitioning from buying products to acquiring services, the burden of integrating different products into a dedicated solution for customers is falling to ecosystem players that need intimate interactions with other players in the ecosystem. However, few, if any, holistic models exist that support organizations to better align their internal and ecosystem dimensions.

In this paper we introduced the EASO model that encompasses elements of both the ecosystem as well as the internal organization. The model provides three perspectives, i.e. strategy, architecture and organizing, and two dimensions, i.e. internal and ecosystem. This provides an approach where the alignment between the perspectives and between the internal and ecosystem dimensions can be analyzed and improved. The development of a new analysis and assessment model including not only the internal organization, but also an external ecosystem dimension is an important improvement of earlier models that primarily focused on either the internal company, e.g. the BAPO model [13], or that only focus on one of the dimensions like the ecosystem, the architecture, or way of organizing.

Through our case examples we show that the new ESAO model is able to incorporate the external and internal strategy, architecture and ways of organizing. Furthermore, the model is applicable not only for product-line companies, but also for other companies that work with software intensive and embedded systems. Finally, the ESAO model is not a sequential model, in which one dimension if followed by another dimension. Instead, the ESAO model focuses on analyzing and achieving alignment between all the different dimensions. The different dimensions of ESAO impact each other and need to be aligned for software intensive organizations in order to be able to develop their organization within their particular ecosystem as well as to react to changes occurring in their ecosystem.

From the case examples it also became clear that firms with different roles in their ecosystem choose different strategies to either maintain their role or re-act to changes in the ecosystem and ecosystem roles. However, the different roles played by firms might have an impact on the alignment of the ESAO components. It might be that different ecosystem roles imply a trade off or prioritization of the other ESAO elements since these different ecosystem roles might have different implications for the architecture, strategy and way of organizing within a firm. Future work would

need to study the implications of working in multiple ecosystems with different roles for the internal and external dimensions.

The proposed model of ESAO offers an analysis and assessment framework, as well as a framework supporting change and development within software intensive organizations. In this paper, the focus is on the analysis, but in future work, we intend to expand on the other uses for the model as well.

References

1. Moore, J.F.: Predators and prey: a new ecology of competition. Harvard Business Rev. 71(3), 75–86 (1993)
2. Moore, J.F.: Business ecosystems and the view from the firm. The Antitrust Bulletin 51(1), 31–75 (2006)
3. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Business Rev. 82(9), 69–78 (2004)
4. Gawer, A., Cusumano, M.: Platform leadership. Harvard Business School Press, Harvard (2002)
5. Gawer, A., Henderson, R.: Platform owner entry and innovation in complementary markets: evidence from Intel. J. of Eco. and Management Strategy 16(1), 1–34 (2007)
6. Li, Y.-R.: The technological roadmap of Cisco's business ecosystem. Technovation 29, 379–386 (2009)
7. Bosch, J., Bosch-Sijtsema, P.M.: From integration to composition: On the impact of software product lines, global development and ecosystems. J. of Systems and Software 83, 67–76 (2010)
8. Manikas, K., Hansen, K.M.: Software Ecosystems –A Systematic Literature Review. J. of Systems and Software 86(5), 1294–1306 (2013)
9. Hofmeister, C., Kruchten, P., Nord, R.L., Obbink, H., Ran, A., America, P.: A general model of software architecture design derived from five industrial approaches. J. of Systems and Software 80(1), 106–126 (2007)
10. Bosch, J., Bosch-Sijtsema, P.: Coordination between global agile teams: From process to architecture. In: Šmite, D., Brede Moe, N., Ågerfalk, P. (eds.) Agility Across Time and Space. Implementing Agile Methods in Global Software Projects, pp. 217–233. Springer, Berlin (2010)
11. Larman, C.: Agile and Iterative Development: A Manager's Guide. Addison-Wesley, Boston (2004)
12. Schwaber, K.: Agile Software Development with Scrum. Prentice Hall (2001)
13. van der Linden, F.J., Dannenberg, R.B., Kamsties, E., Känsälä, K., Obbink, H.: Software product family evaluation. In: Nord, R.L. (ed.) SPLC 2004. LNCS, vol. 3154, pp. 110–129. Springer, Heidelberg (2004)
14. Teece, D.J.: Business models, business strategy and innovation. Long Range Planning 43, 172–194 (2010)
15. Boudreau, K.J., Lakhani, K.R.: How to manage outside innovation. MIT Sloan Management Rev. 50(4), 69–76 (2009)
16. Olsson, H.H., Alahyari, H., Bosch, J.: Climbing the "Stairway to Heaven"–A Multiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In: 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 392–399. IEEE Press (2012)
17. Bosch, J.: Achieving Simplicity with the Three Layer Product Model. IEEE Comp. 46(11), 34–39 (2013)

KPIs for Software Ecosystems: A Systematic Mapping Study

Farnaz Fotrousi[1], Samuel A. Fricker[1], Markus Fiedler[1], and Franck Le-Gall[2]

[1] Blekinge Institute of Technology, Karlskrona, Sweden
{farnaz.fotrousi,samuel.fricker,markus.fiedler}@bth.se
[2] Easy Global Market, Sophia-Antipolis, France
franck.le-gall@eglobalmark.com

Abstract. To create value with a software ecosystem (SECO), a platform owner has to ensure that the SECO is healthy and sustainable. Key Performance Indicators (KPI) are used to assess whether and how well such objectives are met and what the platform owner can do to improve. This paper gives an overview of existing research on KPI-based SECO assessment using a systematic mapping of research publications. The study identified 34 relevant publications for which KPI research and KPI practice were extracted and mapped. It describes the strengths and gaps of the research published so far, and describes what KPI are measured, analyzed, and used for decision-making from the researcher's point of view. For the researcher, the maps thus capture state-of-knowledge and can be used to plan further research. For practitioners, the generated map points to studies that describe how to use KPI for managing of a SECO.

Keywords: software ecosystem, digital ecosystem, performance indicator, KPI, success factor, systematic mapping.

1 Introduction

A software ecosystem (SECO) is about "the interaction of a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationship among them" [1]. We include here any ecosystem that is based on or enabled by software, including pure software, software-intensive systems, mobile applications, cloud, telecommunications, and digital software ecosystems. The inclusion of telecommunications, for example, is important as many modern software services can only be realized with appropriate ICT infrastructure. Companies adopt a SECO strategy to expand their organizational boundaries, to share their platforms and resources with third parties, and to define new business models [2, 3].

A SECO is frequently supported by a technological platform or market that enables the SECO actors in exchanging information, resources, and artifacts. Ownership of such a platform gives strategic advantages over the other SECO actors. It allows satisfying ever-increasing customer demands with limited own resources. It also

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 194–211, 2014.

allows improving one's own knowledge about the marketplace. Such knowledge is necessary for innovation, evolution of a product or service offering, and identification of revenue opportunities [4, 5].

SECO platform ownership also brings responsibilities. These include the definition of SECO performance objectives and management of the SECO to achieve these objectives. A SECO is expected to be healthy [6] and sustainable [7]. It is healthy when it is productive for surrounding actors, robust, and niche-creating [8]. It is sustainable when it maintains its structure and functioning in a resilient manner [6]. Health and sustainability are closely linked performance objectives [9] that are often found in complex systems [10].

Managing a SECO involves definition of how actors, software, and business models play together to achieve the SECO objectives [11] in business, technical, and social dimensional perspectives [12]. The platform owner uses performance indicators for benchmarking and monitoring the resulting ecosystem behavior. Key performance indicators (KPI) are those among the many possible indicators that are important, easily measurable quantitatively or with an approximation of qualitative phenomena [13]. The KPI serve as early warnings about potentially missed SECO objectives [14] and to detect patterns that are useful for predicting health and sustainability of the SECO [15]. Any deviation from success baselines are recorded and acted upon to ensure that the main ecosystem's objectives are met.

The here presented study gives an overview of literature on KPI for software ecosystems. A systematic mapping methodology was followed to identify and classify publications based on the reported research and based on KPI use. The dimensions used for classifying research were the type of ecosystem that was studied and the type of result that was delivered by the research. The dimensions used for classifying KPI use were the researched KPI types, the SECO objectives these KPI were used for.

The knowledge gap for collecting evidences about KPI studies motivated to systematically evaluate distribution of studies and provide guidance for future improvement. For practitioners, the generated map describes how to use KPI in the management of a SECO. It enables the platform owner in understanding the indicators that are important to assess for given SECO objectives. For researchers, the generated map describes state of research and helps finding research gaps for understanding the definition and use of SECO KPI.

The remainder of the paper is structured as follows. Section 2 presents the research objectives and defines research questions, search strategy, study selection, and study quality assessment. Sections 3 and 4 present the results by giving an overview of SECO KPI research, respectively SECO KPI practice. Section 5 discusses the results. Section 6 summarizes and concludes.

2 Research Methodology

The goal of this study is to provide an overview of the research performed to investigate the use of KPI for managing software ecosystems. The systematic mapping approach [16] allows to map the frequencies of publications over categories

to see the current state of research. It also exposes patterns or trends of what kind of research is done, respectively has been ignored so far. Mapping the research results, in addition to the type of research, reveals researchers' current understanding of KPI-related practice.

2.1 Research Questions

To provide an overview on publications relevant to KPI use for SECO, two sets of research questions are defined in Table 1. With the first set of questions we mapped foci and gaps of research about SECO KPI. With the second set we mapped the state of practice that was reported by the research.

Table 1. Research Questions

SECO KPI Research	**Rationale**
RQ1: What kinds of ecosystems were studied?	The answer to this question shows the intensity of SECO KPI research across application domains and types of ecosystems. Skewedness, e.g. due to a focus on just a few types of application domains and ecosystems, indicates gaps where additional research is needed.
RQ2: What types of research were performed?	The answer to this question shows the maturity of SECO KPI research. The more disproportioned conceptual solutions and empirical validation research are, the more there is a need for research that compensates.
Ecosystem KPI Practice	**Rationale**
RQ3: What objectives were KPI used for?	The answer to this question shows the purposes of SECO KPI. It allows understanding when a SECO is considered to be successful and when not. Correlation with the answer to RQ4 allows understanding how the satisfaction of these SECO objectives is measured.
RQ4: What ecosystem entities and attributes did the KPI correspond to?	The answer to this question gives an overview of relevant KPI that are used to assess achievement of SECO objectives. The KPI show how SECO objectives are operationalized and quantified. Skewedness, a focus on just one or a few KPI, may indicate the degree of universality the KPI have for SECO management.

2.2 Systematic Mapping Approach

To answer RQ1, RQ3, we followed the systematic mapping guidelines proposed by Petersen [16]. We (i) conducted database search with a search string that matched our research scope, (ii) performed screening to select the relevant papers, (iii) built a classification scheme based on keywording the papers' titles, abstracts, and keywords, and (iv) used this classification scheme to map the papers. To answer RQ2, we

modified the mapping process by using the pre-existing classification schemes already used in [16, 17]. For RQ4, we built the classification scheme by extracting keywords from the main body of the papers and aligning the emerging scheme with the relevant software industry standard. The research steps are explained below.

(i) Database Search. The study defined the following search strategy.

Search String. To get an unbiased overview of KPI use in SECO, the search string was created with keywords that capture population only. The first aspect used to define the population was the ecosystems that can be found in a software context: software, digital, mobile, service, cloud, telecommunication, and ICT ecosystems. We also included papers that focused on software supply by adding software supply to the search string. The second aspect used to define the population was the application or use of KPI. We used the terms indicators, metrics, measurements, success factors, key characteristics, and quality attributes as synonyms for KPI. To avoid bias about RQ3, we did neither constrain for what purpose information was gathered and used. To build a broad overview of the research area and avoid bias, no keywords were defined in relation to intervention (e.g. monitoring), outcomes (e.g. improvements to a SECO), or study designs (e.g. case studies).

The search string was built by concatenating the two population aspects with the *AND* operator. The search string was formulated as follows: *software OR (software-intensive) OR digital OR mobile OR service OR cloud OR communic* OR telecom* OR ict) PRE/0 (ecosystem* OR "supply network*") AND (measur* OR kpi* OR metric* OR analytic* OR indicator* OR "success factor*" OR "quality attribute*" OR "key characteristic*"*.

Search Strategy. The papers were identified using the important research databases in software engineering and computer science including Scopus, Inspec, and Compendex, which support IEEEXplore and ACM Digital Library as well. The search string was applied to title, author's keywords and abstract of these papers. The search did not restrict the date of the publication.

Validation. We validated the set of identified papers by checking it against the papers used in the SECO literature reviews performed by [2, 5]. Each paper used by these studies that was relevant for our study had been found by following the above-outlined database search.

(ii) Screening of Papers. The inputs for this step were the set of papers identified with step (i). The first and second authors screened these papers independently We screened these papers to exclude studies that do not relate to the use of KPI for any ecosystem-related purpose and to ensure broad-enough coverage of the population. We describe here a complete set of inclusion and exclusion criteria.

Inclusion. We included peer-reviewed journal, conference, or workshop papers that were accessible with full text. The included papers describe the use of KPI in an ecosystem context or the effects of such KPI on properties of the ecosystem. Due to the importance of networking infrastructure and digital information exchange for a well-functioning software ecosystem we included telecommunication and information technology papers in addition to pure SECO papers.

Exclusion. We excluded papers that focused on the use of KPI for managing a member of the ecosystem only. For example, papers about the use of indicators for managing a single company that participates in the ecosystem, or a product or process of that company, were excluded because of their too narrow focus. We excluded papers that focused on other ecosystems rather than a software ecosystem. For example papers focus on biology, environmental, climate, and chemical aspects were excluded. When the definition of software ecosystem did not fulfill in the papers, they were excluded. As an example, the paper that considered Bugzilla and email system as software ecosystems was excluded, since such systems do not address the shared market concept of a SECO definition. Papers that study qualitative indicators using qualitative approaches such as a structured interview were excluded. Also, we excluded papers that focused on ecosystem design in place of ecosystem management. For example, papers about the design of interoperability protocols or of products or services offered to an ecosystem were excluded. The papers that do not Finally, to avoid inclusion of papers that only speculated about KPI use or effects, we excluded papers that did not report any empirically-grounded proof-of-concept.

(iii) Building the Classification Scheme. To answer the research questions RQ1, RQ3, and RQ4 we employed keywording [16] as a technique to build the classification scheme in a bottom-up manner. Extracted Keywords were grouped under higher categories to make categories more informative and to reduce number of similar categories. We built the ecosystem classification scheme by extracting the types and application domains of the studied ecosystems. We built the classification scheme for KPI practice by extracting KPI assessment objectives, entities and attributes used for measuring the KPI.

The keywords were extracted from the papers' titles, keywords, and abstracts. When the quality of an abstract was too poor, we used the main body of the paper to identify the keywords. Similarly, as most of the papers did not included sufficient information about entities and attributes measured with KPI inside the abstract, we used the main body of the papers for keyword identification. The keywords obtained from extraction were then combined and clustered to build the categories used for mapping the papers. The clustering of measurement attributes was aligned with the categories described in ISO/IEC FDIS 25010 as far as applicable.

To answer RQ2, we used a pre-defined classification scheme [17] that was used by earlier systematic mapping studies [16]. It classifies research types into validation research, evaluation research, solution proposals, philosophical papers, opinion papers, and experience papers.

(iv) Systematic Mapping of the Papers. When the classification scheme was in place, the selected papers were sorted into the classification scheme. The classifications were then calculated the frequencies of publications for each category.

To answer RQ1 and RQ2 we reported the frequencies of the selected papers for the categories in the dimensions of ecosystems types and application domains, respectively in the dimensions of research type and research contributes type. We used x-y scatterplots with bubbles in category intersections to visualize the kinds of ecosystems that were studied. The size of a bubble is depicted proportional to the number of papers that are in the pair of categories that correspond to the bubble

coordinates. The visualized frequencies make it possible to see which categories have been emphasized in past research and which categories received little or no attention.

To answer RQ3, we first described the categories identified when building the classification scheme and how these categories were expressed in the selected papers. This description resulted in a dictionary for interpreting the scatterplots used for describing how SECO KPI are used in relation to these objectives. We again used x-y scatterplots for showing the frequency of pairs of categories. These pairs allowed us to describe the attributes measured for each type of ecosystem entity, the measurements used in relation to the SECO objectives, and how KPI are obtained for various kinds of entities found in a SECO.

2.3 Threats to Validity

This section analyzes the threats to validity for the taxonomies of construct, reliability, internal and external validity.

Construct validity reflects whether the papers included in the study reflect the SECO KPI phenomenon that was intended to be researched. The search string was constructed in an inclusive manner so that it captured the wide variety of software-related ecosystems and the many different names given to key performance indicators. The common databases, used for software and management-related literature research, were used to find papers. Only after this inclusive process, manual screening was performed to exclude papers not related to the research objectives. The list of included papers was then validated against two systematic studies on software ecosystem [2, 5] and found that the review covers all relevant papers.

Reliability validity refers to the repeatability of the study for other researchers. The study applied a defined search string, used deterministic databases, and followed a step-by-step procedure that can be easily replicated. The stated inclusion and exclusion criteria were systematically applied. Reliability of the classification was achieved by seeking consensus among multiple researchers.

Internal validity treats refers to problems in the analysis of the data. These threats are small, since only descriptive statistics were used.

External validity concerns the ability to generalize from this study. Generalization is not an aim of a systematic mapping study as only one state of research is analyzed and the relevant body of research completely covered. In particular, the study results about the use of SECO KPI, reflects the practices studied in SECO KPI research and not SECO KPI practice performed in general.

3 Results: Ecosystem KPI Research

The database search resulted in a total of 262 papers, including 46 duplicates. After screening and exclusion, 34 papers remained and were included in the study. These selected papers were published from 2004 onwards. This section gives an overview of the research described in the selected papers. Appendix A lists the selected papers.

3.1 Kinds of Ecosystems

To answer RQ1, Figure 1 gives an overview over the ecosystems that our study found KPI research for. The number embedded in a bubble indicates how many papers were devoted to a given combination of ecosystem type and application domain (multiple classifications possible). Empty cells indicate that no corresponding study was found. The number on the category label indicates the total number of papers in that category.

Most of the papers used the term software ecosystem to characterize the studied ecosystems. Special kinds of ecosystems were cloud, service, mobile apps, and open source software ecosystems. Less frequent were digital ecosystems with 44% of the papers. They refer to the use of IT to enable collaboration and knowledge exchange [18].

The papers addressed a variety of application domains. Most common were telecommunications, business management and software development. None of the remaining application domains was addressed by more than one or two papers. Thus research is rather scattered, and the specifics of the various application domains only little understood.

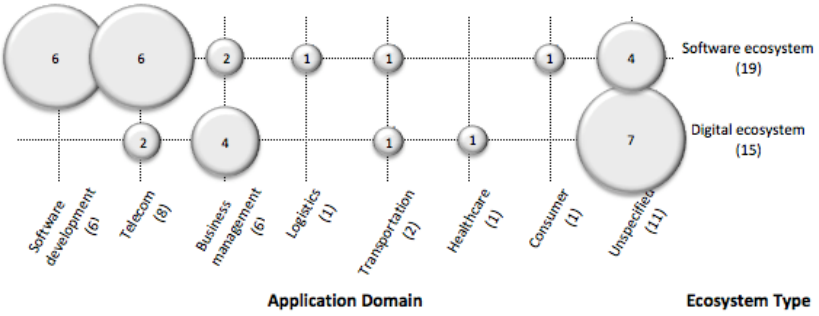

Fig. 1. Kinds of ecosystems that were studied with KPI research. The label "software ecosystem" refers to those that are not considered a digital ecosystem (see main text).

3.2 Types of Research

To answer RQ2, Figure 2 presents a map of the kind of research performed on KPI in software-related ecosystems. Papers with multiple research types and contributions were classified for each combination of research type and contribution they presented.

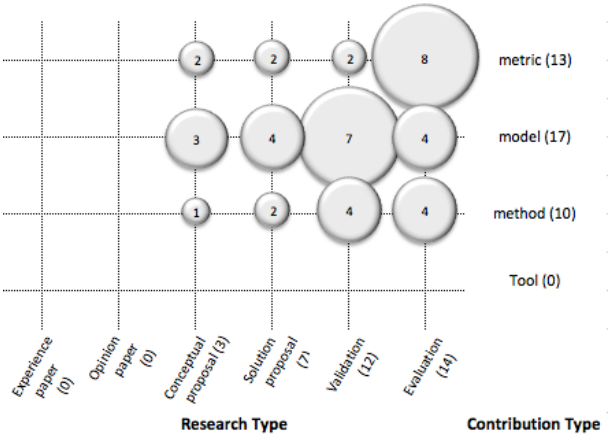

Fig. 2. Map of research on SECO KPI and type of contributions

Experience report papers describe experiences in working with SECO KPI and usually describe unsolved problems. *Opinion papers* discuss opinions of the papers' authors. *Conceptual proposal* papers sketch new conceptual perspectives related to SECO KPI. This category renamed *philosophical papers* category (described in iii of section 2.2) to fit the SECO KPI study. *Solution proposal* papers propose new techniques or improve existing techniques using a small example or a good argumentation. *Validation* papers investigate novel solutions that had not been implemented in practice (e.g. experiment, lab working). *Evaluation* papers report on empirical or formal studies performed to implement a solution or evaluate the implementation.

Metric papers describe KPI for SECO. *Model* papers describe relationships between KPI. *Method* papers describe approaches for working with SECO KPI. Finally, *tool* papers describe support for work with SECO KPI.

Most research was found in the categories of validation and evaluation. Research contributed with metrics, models, or methods. For example, R17 proposes a model that explains how health can be measured with relevant indicators (conceptual proposal, model) and validates that model with a questionnaire (validation, model). R14 proposes a method for assessing services based on Quality of Service indicators (solution, method). R19 evaluates factors that affect successful selling in e-markets (metric, evaluation). No paper was an experience report or an opinion paper. No paper contributed with any tool.

4 Results: Researched KPI Practice

The papers included in this study describe the use of KPI by a platform owner for achieving objectives with the ecosystem that was enabled by the ecosystem platform. This section gives an overview of these objectives and the KPI that were used.

4.1 Ecosystem Objectives Supported by KPI

KPI were used to enable or achieve a variety of objectives. Platform owners aimed, at improving business, at improving the interconnectedness between actors, at growing the ecosystem, at improving quality of ecosystem, product, or services performed within the ecosystem, and at enabling sustainability of the ecosystem (answer RQ3):

Business improvement. Research has been performed on how to improve business at the ecosystem level. The studied business improvements concerned the perspectives of ecosystem activity and of commercial success. Ecosystem activity related to the level of activity of participating actors, encouragement to participate in the ecosystem, and the transaction volume. Commercial success related to sales success, innovativeness and competitiveness of the participating actors, and the cost of the network that enables the ecosystem. The activity and commercial perspectives were mixed in the papers, thus could not be separated in the analysis of the literature.

Interconnectedness improvement. Research has been performed on how to improve interaction in an ecosystem, for example to reduce cost, improve predictability of services that are provided in the ecosystem, and manage trust. Interaction improvement was studied between individual actors and between whole networks contained in the ecosystem. The research differed in terms of lifecycle stage of an interaction and covered supplier availability, discovery, ranking and selection, the resulting connectivity, interaction evaluation, and the impact of the interaction on the actors that participated in it. Interaction improvement was not always an end in itself, but was considered essential for generating business activity and sustainability of the ecosystem.

Growth and stability. Research has been performed on how to manage growth and stability of the ecosystem. Growth and stability were seen as two factors that need to be managed jointly. During growth flexibility and controllability need to be maintained. During stability, a continuous co-revolution must happen. Growth and stability again are not ends in themselves, but thus contribute to sustainability and survival of the ecosystem.

Quality improvement. Research has been performed on how to manage quality of ecosystems. In particular, performance, usability, security, data reliability, extendibility, transparence, trustworthiness, and quality-in-use were investigated. Quality management was sometimes presented as an ends in itself, for example by allowing comparison among multiple ecosystems, enabling diagnosis, improving decision-making, and achieving long-term usage of services. At the same time, however, quality management was considered to be a means to encourage adoption and growth, improve business performance, and achieve sustainability.

Enable sustainability. Research has been performed on how to sustain an ecosystem. Two angles were taken: self-organization and resource consumption. Self-organization was approached through continuous rejuvenation of the ecosystem. Resource consumption was studied in relation of electrical energy. Throughout all papers found in this category, sustainability was considered to be desirable ends for software ecosystems.

4.2 KPI: Measured Entities

The included papers describe measurements applied to the ecosystem as a whole as to the parts the ecosystem consists of: actor, artifact, service, relationship, transaction and network.

Actors. Actors were measured and characterized as follows. They were human or artificial. Examples of human or legal actors were sellers and developers that provide products to buyers or groups of organizations and firms. Examples of artificial actors were nodes in a telecommunication network. An actor engages in transactions in an ecosystem and builds relationships to other actors or artifacts. The transactions the seller engages in generate profit and revenue for the cost the seller is willing to take. Effective actors have knowledge about other actors or the network and has good interestingness and reputation for other actors. Actors are also considered to be sources and sinks of data and have differing ranges for data transmission. Performance of individuals and groups in terms of fulfilled tasks and decisions as well as performance of firms and organizations in terms of profits are measured.

Artifacts. Artifacts such as software, codes, plugins, books, music, or data were measured and characterized as follows. Artifacts had a location in the ecosystem. They evolve, may have reputation and popularity, and exposed their consumers to vulnerability.

Services. Services were measured and characterized as follows. Services consume energy and other resources. Services have quality attributes such as quality of service, security, compliance, and reputation. Metadata and service level agreements are used to specify the services. The services are not fixed but evolve: services emerge, change, and get extinct. A special service was provided by the platform that laid the fundament for the ecosystem. It was characterized in terms of attributes like stability, documentation, portability, and openness.

Relationship. Relationships were measured and characterized as follows. Actors enter relationships with other actors, artifacts, or services. A relationship connects two or more such entities. Examples of relationships were business connections and telecommunication communication links. A relationship may be transparent and express a trust value of the connected entities. A relationship is the basis for transactions, thus is used for advertising and building alliances. The transaction, however, is constrained by cost and quality of the relationship.

Transactions. Transactions were measured and characterized as follows. Examples of transactions are sales of services to customers, server requests, and commits of code files made by developers. They are initiated with an offer that is measured in terms of attributes like price and quantity. Transactions also have a price and quantity. Other attributes include time to negotiate the transaction, time to complete, energy consumption, transmission rate, and buyer satisfaction.

Network. Networks were considered as sets of entities and relationships that were part of a whole ecosystem. Examples were local or application-specific networks. Networks were characterized as follows. Networks were vulnerable to security threats such as data availability, integrity, authentication, and authorization. Networks differed in the node density, degree of collaboration, provisioning cost, and hit rate for artifacts.

Ecosystem. Full ecosystems were characterized as follows. They have quality attributes like size, performance, security and energy consumption that can also characterize networks contained in an ecosystem. In addition, ecosystems exhibited lifelines, diversity, stability, transparency, healthiness, and sustainability.

This section and next section collaboratively provide answer for RQ4. The map in the left part of Figure 3 shows the entities that were studied in relation to the ecosystem objectives. Most research studied the measurement of the overall ecosystem to enable quality or business improvement. For example, R17 describes how performance of the ecosystem affected user satisfaction, and R13 shows how analytics applied to the ecosystem can be used to improve business. Considerable research was also devoted to improving the interconnectedness of the ecosystem, where attributes of the products and services played an important role and also to the role of platform measurements to grow the ecosystem and improve quality. For example, R6 described how to use a service similarity measurement was used to improve ecosystem connectivity. R2 described how growth, diversity, and entropy measurements of a SOA platform were used to increase growth. R4 described how communication quality measurements were used to improve the quality of a telecommunication ecosystem.

The map also shows areas where no research was published. For example no research studied the role of network measurements for objectives other than sustainability and quality improvement.

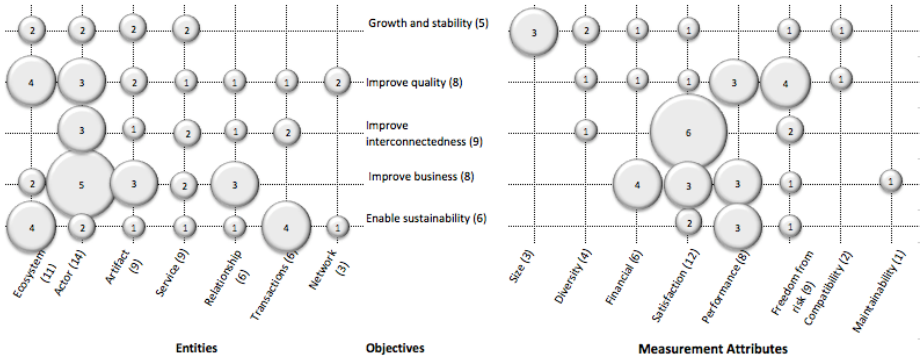

Fig. 3. Map of measured entities and measurement attributes in relation to ecosystem objectives

4.3 KPI: Measurement Attributes

To make the state and evolution of the ecosystem and of its elements visible, a broad variety of attributes were measured.

The following attributes categories emerged when clustering the attributes described in the included papers. Figure 4 shows how classes of quality attributes were merged toward new categories. The *size* category includes attributes to measure size and growth. *Diversity* includes attributes to measure heterogeneity and openness for such heterogeneity. *Financial* includes attributes to measure economic aspects

such as investment, cost, and price. *Satisfaction* includes attributes to measure satisfaction and the related concepts of suitability, interestingness, learnability, usability, accessibility, acceptability, trust, and reputation. *Performance* includes attributes to measure performance, including resource utilization, efficiency, accuracy, and effectiveness. *Freedom from risk* includes attributes to measure the ability to avoid or mitigate risks and includes the related concerns of security, reliability, maturity, availability, and other related guarantees. *Compatibility* includes attributes to measure the degree to which an entity can perform well in a given context, interoperate or exchange information with other entities, and be ported from one context to another one. *Maintainability* includes attributes to measure flexibility, respectively the ability to be changed.

The right part of Figure 3 gives an overview of the attributes referred to by KPI. Most research studied measurements of satisfaction, typically to improve business or interconnectedness. An example of such research is R13 that describes the use of seller reputation to improve business. To support quality improvement, all measurement attributes that relate to quality were included in at least one research paper, except for maintainability and size. Similarly, size measurements did not play any role other than for growth and stability.

The left part of Figure 5 shows how the ecosystem elements were measured. Satisfaction was a common attribute that was measured for any entity except for rules. This shows that a same attribute can be measured or analyzed for different ecosystem entities. Also it is revealed that similar measurement attributes might be collaborating to measure different ecosystem elements. As an example CCCI (correlation, commitment, clarity and importance) measurable attributes were used to measure trust as well as reliability.

- *Diversity*
  - Heterogeneity
  - Openness
- *Satisfaction*
  - Satisfaction
  - Suitability
  - Interestingness
  - Learnability
  - Usability
  - Accessibility
  - Acceptability
  - Trust
  - Reputation.
- *Performance*
  - Performance
  - Resource utilization
  - Efficiency
  - Accuracy
  - Effectiveness
- *Financial*
  - Investment
  - Cost
  - Price
- *Size*
  - Size
  - Growth
- *Freedom from risk*
  - Risk mitigation
  - Security
  - Reliability
  - Maturity
  - Availability
  - Guarantees.
- *Compatibility*
  - Interoperability
  - Exchangeability
- *Maintainability*
  - Flexibility
  - Changeability

Fig. 4. Merging classifications of measurement attributes

The overall ecosystem was the most comprehensively measured or analyzed entity, with a special focus on satisfaction, freedom from risks and performance. Some examples of such satisfaction measurements are provided by R13 that measured usage and acceptability of an ecosystem. The platform followed with the second-largest variety of measurements. R2, for example, measured entropy and diversity to characterize platform complexity. Only narrow sets of measurement attributes were applied to the business partner, interactions, and business.

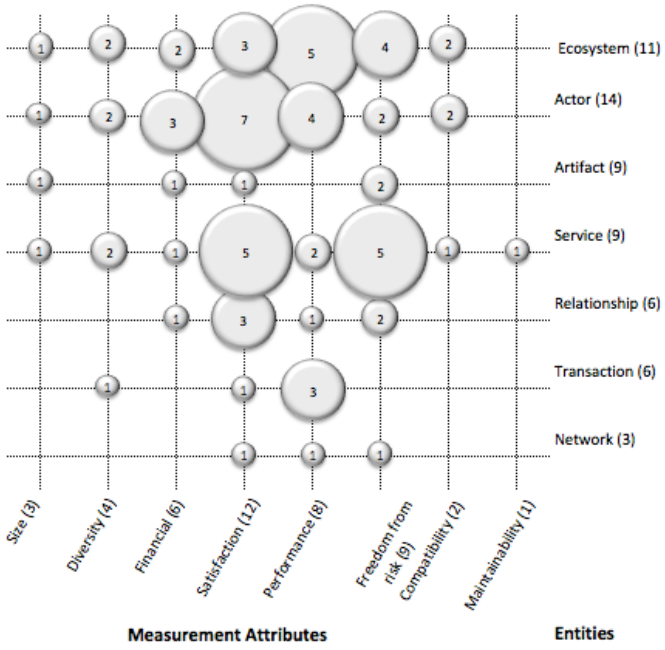

Fig. 5. Map of measurement attributes in relation to the measured entities

5 Discussion

The study provides a classification of KPI relevant papers in understanding researches, relationship with the practice, and assessment of research outcomes. This classification contributes to taxonomy, which can help for closer examination of the ecosystem or platform owner objectives, making them more recognizable in designing KPI. New KPI can be extracted for an ecosystem using this taxonomy, and existing KPIs can be extended or restructured applying the generic structure of the taxonomy.

The literature map indicates that KPI for software-based ecosystems is a thin area with work at all maturity levels. Journal, conference, and workshop papers exist. However, the number of publications is not sufficient, and many application domains for ecosystems addressed with just one or two papers. Although formulation of KPI might be domain dependent and similarity of objectives is not the only factor to select

a KPI, however due to insufficient study it is difficult to state whether characteristics of a domain, for example regulation of healthcare, affects the KPI of the ecosystem that targets that domain.

The included research on ecosystem KPI mostly addresses ecosystem measurements or measurements of satisfaction, performance and freedom from risks. Measurements other than satisfaction that are applied on elements contained in the ecosystem are comparatively little researched. A broader understanding of KPI would increase a platform owner's flexibility in measuring, analyzing, and using KPI for decision-support. The understanding of a greater variety of KPI would also contribute to increased transparency of status, evolution, and other aspects of the ecosystem.

6 Conclusion

The here presented study gives an overview of literature on the use of KPI for software-based ecosystems. A systematic mapping methodology was followed and applied to 34 included studies published from 2004 onwards.

To respond to RQ1 and RQ2, research was broad but thin. Two major kinds of ecosystems were researched: software ecosystems and digital ecosystems. Many application domains were addressed, but most of them with one or two papers only. The published research was mature with journal, conference, and workshop papers that covered metrics, models, and methods. In response to RQ3 and RQ4, KPI research was skewed. Most research studied ecosystem KPI for improving the interconnectedness between individual actors and subsystems of the ecosystem. Overall, most KPI were about satisfaction, performance and freedom from risks measures.

The results of the mapping study indicate that more research is needed to better understanding of KPI for software-based ecosystems. In particular, a deeper understanding of how the application domain affects an ecosystem's KPI is needed. Also, an important research opportunity is the identification, analysis, and evaluation of KPI. Such research could make the work with KPI more flexible, because a greater variety of KPI would be known and available for the practitioner to use.

References

1. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: International Conference on Software Engineering. ICSE-Companion 2009, Vancouver, British Columbia, Canada (2009)
2. Manikas, K., Hansen, K.M.: Software ecosystems–a systematic literature review. Journal of Systems and Software 86, 71–80 (2012)
3. Weiblen, T., Giessmann, A., Bonakdar, A., Eisert, U.: Leveraging the software ecosystem: Towards a business model framework for marketplaces. In: 3rd International Conference on Data Communication Networking, DCNET 2012, 7th International Conference on e-Business, ICE-B 2012 and 3rd International Conference on Optical Communication Systems, OPTICS 2012, Rome, Italy (2012)

4. Bosch, J.: From software product lines to software ecosystems. In: 13th International Software Product Line Conference, Carnegie Mellon University (2009)
5. Barbosa, O., Alves, C.: A systematic mapping study on software ecosystems. In: 2nd ICSOB, Brussels, Belgium (2011)
6. Costanza, R., Mageau, M.: What is a healthy ecosystem? Aquatic Ecology 33, 105–115 (1999)
7. Chapin Iii, F.S., Torn, M.S., Tateno, M.: Principles of ecosystem sustainability. American Naturalist, 1016–1037 (1996)
8. Iansiti, M., Richards, G.L.: Information Technology Ecosystem: Structure, Health, and Performance. The Antitrust Bull. 51, 77–110 (2006)
9. Rapport, D.J., Costanza, R., McMichael, A.J.: Assessing ecosystem health. Trends in Ecology & Evolution 13, 397–402 (1998)
10. Costanza, R.: Toward an operational definition of ecosystem health. Ecosystem health: New goals for environmental management, pp. 239–256. Island Press (1992)
11. Manikas, K., Hansen, K.M.: Reviewing the Health of Software Ecosystems–A Conceptual Framework Proposal. In: International Workshop on Software Ecosystems 2013-IWSECO 2013, Potsdam, Germany (2013)
12. Santos, R., Werner, C., Barbosa, O., Alves, C.: Software ecosystems: trends and impacts on software engineering. In: 26th Brazilian Symposium on Software Engineering (SBES), Piscataway, NJ, USA (2012)
13. Parmenter, D.: Key performance indicators (KPI): developing, implementing, and using winning KPIs. John Wiley & Sons (2010)
14. GAO: Performance Measurement and Evaluation: Definitions and Relationships. US Government Accountability Office (2011)
15. Cokins, G.: Performance management: Integrating strategy execution, methodologies, risk, and analytics. John Wiley & Sons (2009)
16. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering, Bari, Italy (2008)
17. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Engineering 11, 102–107 (2006)
18. Boley, H., Chang, E.: Digital ecosystems: Principles and semantics. In: Inaugural IEEE-IES Digital EcoSystems and Technologies Conference, Cairns, Australia (2007)

7 Appendix I: The Selected Studies

ID	References
R1	Sabry, N., Krause, P.: A digital ecosystem view on cloud computing. 6th IEEE International Conference on Digital Ecosystems Technologies (DEST). Piscataway, NJ, USA (2012)
R2	Fiegler, A., Dumke, R.R.: Growth-and Entropy-Based SOA Measurement: Vision and Approach in a Large Scale Environment. Software Measurement,Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA). Los Alamitos, CA, USA (2011)
R3	Pranata, I., Skinner, G., Athauda, R.: TIDE: Measuring and evaluating trustworthiness and credibility of enterprises in digital ecosystem. International Conference on Management of Emergent Digital EcoSystems. San-Francisco, USA (2011)
R4	Yang, Y., Xu, Y., Li, X., Chen, C.: A loss inference algorithm for wireless sensor networks to improve data reliability of digital ecosystems. Industrial Electronics, IEEE Transactions on 58, 2126-2137 (2011)
R5	Savola, R.M., Sihvonen, M.: Metrics driven security management framework for e-health digital ecosystem focusing on chronic diseases. International Conference on Management of Emergent Digital EcoSystems. Addis Ababa, Ethiopia (2012)
R6	Dong, H., Hussain, F.K., Chang, E.: A service concept recommendation system for enhancing the dependability of semantic service matchmakers in the service ecosystem environment. Journal of Network and Computer Applications 34, 619-631 (2011)
R7	Barolli, L., Yang, T., Mino, G., Durresi, A., Xhafa, F.: A simulation system for WSNs as a Digital Eco-System approach considering goodput metric. 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST). Dubai, United Arab Emirates (2010)
R8	Nankani, E., Simoff, S., Denize, S., Young, L.: Enterprise university as a digital ecosystem: Visual analysis of academic collaboration. 3rd IEEE International Conference on Digital Ecosystems and Technologies, DEST'09. Istanbul, Turkey (2009)
R9	Fabregues, A., Madrenas-Ciurana, J., Sierra, C., Debenham, J.: Supplier performance in a digital ecosystem. 3rd IEEE International Conference on Digital Ecosystems and Technologies, DEST'09. Istanbul, Turkey (2009)
R10	van den Berk, I., Jansen, S., Luinenburg, L.: Software ecosystems: a software ecosystem strategy assessment model. Fourth European Conference on Software Architecture. ACM, Copenhagen, Denmark (2010)
R11	Taghizadeh, M., Plummer, A., Aqel, A., Biswas, S.: Towards optimal cooperative caching in social wireless networks. Global Telecommunications Conference (GLOBECOM). IEEE, Miami, Florida, USA (2010)
R12	Dong, H., Hussain, F.K., Chang, E.: Semantic service retrieval and QoS measurement in the digital ecosystem environment. International Conference on Complex, Intelligent and Software Intensive Systems (CISIS). Krakow, Poland (2010)

ID	References
R13	Tian, C.H., Cao, R.Z., Zhang, H., Li, F., Ding, W., Ray, B.: Service analytics framework for web-delivered services. International Journal of Services Operations and Informatics. 4, 317--332 (2009)
R14	Chen, W., Chang, E.: A method for service quality assessment in a service ecosystem. International Conference on Digital Ecosystems and Technologies Inaugural IEEE. Piscataway, NJ, USA (2007)
R15	Koendjbiharie, S., Koppius, O., Vervest, P., van Heck, E.: Network transparency and the performance of dynamic business networks. 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST). Dubai, United Arab Emirates (2010)
R16	Jansen, S.: How quality attributes of software platform architectures influence software ecosystems. International Workshop on Ecosystem Architectures. Saint Petersburg, Russian Federation (2013)
R17	Salem, A.M.B.H., Ghadhab, B.B.: Performance Measurement practices in Software Ecosystem. International Journal of Productivity and Performance Management. 62, 514 - 533 (2013)
R18	Goeminne, M., Mens, T.: A framework for analysing and visualising open source software ecosystems. Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE). Antwerp, Belgium (2010)
R19	Pereira, A., Duarte, D., Meira Jr, W., Góes, P.: Assessing success factors of selling practices in electronic marketplaces. International Conference on Management of Emergent Digital EcoSystems. Lyon, France (2009)
R20	Dong, H., Hussain, F.K., Chang, E.: A QoS-based service retrieval methodology for digital ecosystems. International Journal of Web and Grid Services 5, 261-283 (2009)
R21	Fachrunnisa, O., Hussain, F.K.: A methodology for maintaining trust in industrial digital ecosystems. IEEE Transactions on Industrial Electronics 60, 1042-1058 (2013)
R22	La, H.J., Kim, S.D.: A model of quality-in-use for service-based mobile ecosystem. 1st International Workshop on the Engineering of Mobile-Enabled Systems (MOBS). IEEE, San Francisco, CA, USA (2013)
R23	Ion, M., Danzi, A., Koshutanski, H., Telesca, L.: A peer-to-peer multidimensional trust model for digital ecosystems. 2nd IEEE International Conference on Digital Ecosystems and Technologies (DEST). IEEE, Phitsanuloke, Thailand (2008)
R24	Enokido, T., Aikebaier, A., Takizawa, M.: An integrated power consumption model for communication and transaction based applications. International Conference on Advanced Information Networking and Applications (AINA). Biopolis, Singapore. IEEE (2011)
R25	Wright, J.L., McQueen, M., Wellman, L.: Analyses of two end-user software vulnerability exposure metrics (extended version). Information Security Technical Report 17, 173-184 (2013)
R26	Böhmer, M., Ganev, L., Krüger, A.: Appfunnel: A framework for usage-centric evaluation of recommender systems that suggest mobile applications. International conference on Intelligent user interfaces. ACM, Santa Monica, CA, USA (2013)

ID	References
R27	Eklund, U., Bosch, J.: Architecture for embedded open software ecosystems. Journal of Systems and Software - Article in Press (2014)
R28	Zhang, J., Liang, X.J.: Business ecosystem strategies of mobile network operators in the 3G era: The case of China Mobile. Telecommunications Policy 35, 156-171 (2011)
R29	Walden, J., Doyle, M., Lenhof, R., Murray, J., Plunkett, A.: Impact of plugins on the security of web applications. 6th International Workshop on Security Measurements and Metrics. ACM, Bolzano-Bozen, Italy (2010)
R30	Straub, D., Rai, A., Klein, R.: Measuring firm performance at the network level: A nomology of the business impact of digital supply networks. Journal of Management Information Systems 21, 83-114 (2004)
R31	Vasilescu, B., Serebrenik, A., Goeminne, M., Mens, T.: On the variation and specialisation of workload-A case study of the Gnome ecosystem community. Empirical Software Engineering - Article in Press (2013)
R32	Luna, J., Ghani, H., Vateva, T., Suri, N.: Quantitative Assessment of Cloud Security Level Agreements: A Case Study. 7th International Conference on Security and Cryptography. SECRYPT. INSTICC Press, Setubal, Portugal (2012)
R33	van Angeren, J., Blijleven, V., Jansen, S.: Relationship intimacy in software ecosystems: a survey of the dutch software industry. International Conference on Management of Emergent Digital EcoSystems. ACM, San Francisco, CA, USA (2011)
R34	Liu, Y., Fan, Y., Huang, K.: Service Ecosystem Evolution and Controlling: A Research Framework for the Effects of Dynamic Services. International Conference on Service Sciences (ICSS). IEEE, Shenzhen, China (2013)

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.

Keywords: 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

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 212–226, 2014.

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 access to the program is rather easy but in order to advance on to higher levels

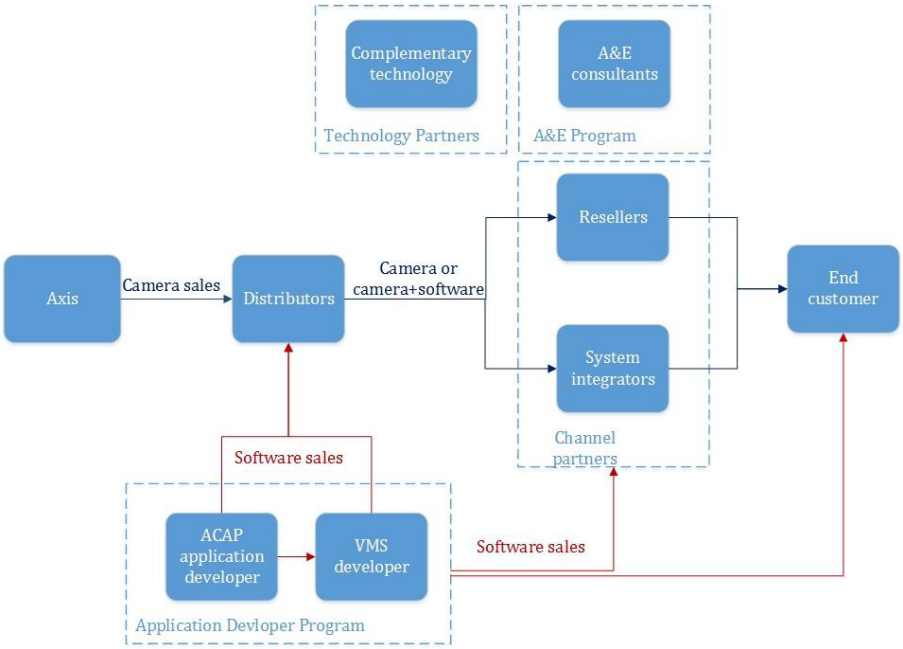

Fig. 1. The software ecosystem surrounding the ACAP, also published in [27]

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 products, 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 relatively 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

although 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 (2013) (Incorporated)
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: 31st International Conference on Software Engineering - Companion, ICSE-Companion 2009, pp. 187–190 (2009)
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), 1495–1510 (2012)
5. Chulani, S., Williams, C., Yaeli, A.: Software development governance and its concerns. In: Proc. of the 1st International Workshop on Software Development Governance, pp. 3–6. ACM, New York (2008)
6. Bannerman, P.L.: Software development governance: A meta-management perspective. In: Proc. of the 2009 ICSE Workshop on Software Development Governance. SDG 2009, pp. 3–8. IEEE Computer Society, Washington, DC (2009)
7. Boehm, B.: A process framework for system and software development governance. In: Proc. of the 1st International Workshop on Software Development Governance. SDG 2008, p. 1. ACM, New York (2008)
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 2009, pp. 9–14. IEEE Computer Society, Washington, DC (2009)
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 2008, pp. 25–28. ACM, New York (2008)

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 2008, pp. 17–20. ACM, New York (2008)
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 2009, pp. 36–39. IEEE Computer Society, Washington, DC (2009)
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 2009, pp. 29–35. IEEE Computer Society, Washington, DC (2009)
13. Ambler, S.W.: Scaling agile software development through lean governance. In: Proc. of the 2009 ICSE Workshop on Software Development Governance. SDG 2009, pp. 1–2. IEEE Computer Society, Washington, DC (2009)
14. Qumer, A.: Defining an integrated agile governance for large agile software development environments. In: Concas, G., Damiani, E., Scotto, M., Succi, G. (eds.) XP 2007. LNCS, vol. 4536, pp. 157–160. Springer, Heidelberg (2007)
15. Baars, A., Jansen, S.: A framework for software ecosystem governance. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 168–180. Springer, Heidelberg (2012)
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.C.: Software development under bounded rationality and opportunism. In: ICSE Workshop on Software Development Governance, pp. 15–20 (2009)
18. Manikas, K., Hansen, K.M.: Reviewing the health of software ecosystems – a conceptual framework. In: 5th International Workshop on Software Ecosystems (IWSECO), pp. 33–44 (2013)
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, ECSA 2010, pp. 127–134. ACM Press, New York (2010)
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, IWSECO 2009, pp. 34–48 (2009)
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 2011, pp. 68–75. ACM, New York (2011)
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/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 2009, Pittsburgh, PA, USA, Carnegie Mellon University, pp. 111–119 (2009)
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 (2014), http://serg.cs.lth.se/index.php?id=89149
28. Manikas, K., Hansen, K.M.: Software ecosystems – a systematic literature review. Journal of Systems and Software 86(5), 1294–1306 (2013)
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), 1455–1466 (2012)
32. Chesbrough, H.W.: Open Innovation: The new imperative for creating and profiting from technology. Harvard Business School Press, Boston (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: Regnell, B., van de Weerd, I., De Troyer, O. (eds.) ICSOB 2011. LNBIP, vol. 80, pp. 32–43. Springer, Heidelberg (2011)
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. Schuur, H.v.d., 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, pp. 76–84. ACM (2011)
38. Bucklin, L.P., Sengupta, S.: Organizing successful co-marketing alliances. Journal of Marketing 57(2), 32–46 (1993)
39. Das, T.K., Teng, B.-S.: Trust, control, and risk in strategic alliances: An integrated framework. Organization Studies 22(2), 251–283 (2001)

App Store Models for Enterprise Software: A Comparative Case Study of Public versus Internal Enterprise App Stores

Stefan Wenzel

Otto-Friedrich-Universität Bamberg, An der Weberei 5, 96047 Bamberg, Germany
SAP AG, Dietmar-Hopp-Alle 16, 69190 Walldorf, Germany
stefan.wenzel@sap.com

Abstract. Mobile app stores have changed the way how consumers discover and buy private software. Employees and end users in companies expect a similar experience and flexibility from their corporate IT. Enterprise software vendors (ESVs) therefore create new modular applications and establish their own versions of app stores for companies. Two models have appeared on the market: the public and the internal enterprise app store (EAS). Public EASs are managed and operated by large ESVs serving them as sales and distribution channels for their software and the software built by their ecosystem. Internal EASs are managed and operated by corporate IT departments to distribute applications to company-internal users. We conduct a comparative case study of one public and one internal EAS and derive recommendations for corporate use to better meet the expectations of today's business stakeholders.

Keywords: App Store, Enterprise Application Software, Business Applications, App'ification, Enterprise App Store, IT Governance, IT Consumerization.

1 Introduction

Enterprise application software, such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM), is traditionally sold via a highly consultative, personnel-intensive process [1]. A customer's software acquisition process is usually governed by a central IT department [2]. Buying cycles of several months up to years are still widely common [3, 4] and significant resources are tied up on the sales and buying side. This is a costly process for both the enterprise software vendor (ESV) and the customer.

Moreover, from an innovation perspective ESVs struggle with early market adoption of newly introduced software products. It is widely accepted in innovation management literature that a new product or invention only classifies as an innovation if it is adopted by a customer [5]. The innovativeness of a software company should therefore not only be measured in terms of "time-to-market", but also in terms of "time from availability to adoption". This laborious go-to-market model also has consequences for the software buying company's internal innovation process:

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 227–242, 2014.

business units often cannot justify the business case for single requirements and IT departments are overextended with consolidating the different needs in the organization or are tied up with operating complex IT landscapes, resulting in innovation bottlenecks [6, 7].

The relationship between business users and IT departments is further complicated by a trend referred to as "consumerization of IT" [8]. Consumer technologies such as smartphones and app stores are pervasive in many people's lives. Hence, business users are nowadays much more knowledgeable and sensitive towards technology in general, but also towards corporate IT and information systems (IS) [9]. Business users ask for IT solutions, with consumer-grade usability, supporting ad-hoc use cases and request a stronger involvement in the software selection process or want to directly select the software they use themselves. Since IT departments, and available enterprise software and the related go-to-market process, cannot comply with these requirements, the role of the CIO or the IT department is questioned [10, 11]. Another consequence is the rise of "shadow IT" [12, 13]: business users circumvent corporate IT rules and use their private devices and applications without permission in their day-to-day work.

ESVs seem to have recognized the described multifaceted dilemma: they are building new, modular ("app-like") applications with consumer-oriented user interfaces [14] and pursue new go-to-market and software distribution models by introducing their own version of app stores[1] for companies, trying to reproduce the success of app stores in the consumer market. These B2B online sales channels not only shift the software acquisition and distribution process from the "offline" to the "online" world, but also promise a change in the enterprise software adoption paradigm: they favor a business-driven bottom-up approach over the traditional IT-driven top-down approach [9]. These EASs are referred to as "public EASs" and are usually managed by a software provider.

With mobile apps entering the enterprise and the need for mobile application management (MAM), another form of EAS has emerged: the internal EAS[2]. In contrast to the public EAS, the internal EAS is managed by the individual software customer company.

By introducing EASs, researchers expect to effectively counter the problems arising with IT consumerization and shadow IT by satisfying the needs of the business users, while gaining back control of the IT used in the company [13]. Software vendors hope to benefit from the app store model with a reduction in cost of sales, increased reach to business users and an acceleration of adoption rates of new products [15]. However, software customers seem to be reluctant and adoption rates of EASs are still low [16, 17]. One reason might be the uncertainty of how to best use these new models in the corporate context, i.e., how to integrate enterprise app stores into "sourcing, delivery and support" of corporate IT services [2].

---

[1] Examples are: SAP Store [19], Salesforce.com AppExchange [43], Microsoft Pinpoint [44], Amazon AWS Marketplace [45], Google Apps Marketplace [46].

[2] Examples are: SAP Enterprise Store [20], Symantec App Center [47], App47 [48], Salesforce.com Private AppExchange [49], OpenPeak Openshop [50]

Therefore, the research objective of this study is to investigate the two prevailing models of EASs: the public and the internal EAS. The individual use cases are evaluated from a software customer's perspective, the differences of each model are highlighted and the respective consequences are discussed. Furthermore, we propose to combine the two models to leverage the advantages of both EASs.

To pursue the research objectives an explorative, qualitative research strategy has been chosen, using an idiographic and comparative case study design [18]. SAP serves as organizational case study context, since it provides a public EAS, the SAP Store [19], and an internal EAS, the SAP Enterprise Store [20]. The article targets IS researchers interested in the under-investigated topic of EASs and their implications towards corporate IT processes as well as companies evaluating the use of EASs or those looking for new ways to provide corporate IT to business units and users.

The article is structured as follows. After a comprehensive literature review on related fields of research, the research methodology is presented. Chapter 3 presents the two cases: the public SAP Store and the internal SAP Enterprise Store. Thereafter the individual findings are compared and discussed. Based on the outcome of the case study a framework is derived how to combine the two models. The work concludes with a discussion on limitations of the study, a summary of the findings and a market and research outlook.

2 Related Work

In this section contributions from multiple streams of research are presented which help to explain the novel socio-technological context of EASs or to assess it. The works are at the crossroads of IS research and Industrial Marketing.

IT Consumerization. As described previously, IT consumerization refers to the trend that IT innovations are adopted first by consumers and are subsequently diffused into enterprise segments [9]. The widespread use of consumer technologies lets business users rate corporate IT and IS with the eyes of a consumer: simple and visually appealing user interfaces, instant or ad-hoc use, and self-determined selection of software are exemplary expectations of business users towards corporate IT. These expectations are often not addressed with today's corporate IS and governance models [9]. Therefore the diffusion of consumer technology into companies is often driven by individual employees and not controlled or permitted by the IT department ("infiltration") [8]. To provide evidence of this phenomenon, Harris et al. (2012) present a survey among 4017 employees: 52% responded that they would at least sometimes use their personal consumer devices for work-related activities, 36% stated that they would not worry about IT policies in place and just use the technology they need to perform their work, and 45% agreed with the statement that private devices and software applications are more useful than the ones provided by corporate IT [21]. Furthermore, Harris et al. (2012) identify three major benefits of IT consumerization for companies: increased innovativeness, productivity, and employee satisfaction. If IT consumerization is not managed actively in the company it leads to shadow IT [13] and its risks typical IT targets such as data security, reliability, and

integrity [21]. Therefore, it is proposed to actively manage IT consumerization or to introduce new governance models such as "Bring your own device" (BYOD, [8]). Beimborn and Palitza (2013) mention EASs as a promising option to manage consumerization and counteract shadow IT [13].

IT Governance. Meyer et al. (2003) define IT governance as "a set of principles, practices, and measures to ensure corporate targets are met with the used IT, while resources are used responsibly and risks are adequately monitored" (translated from original German version, [2]). From a process perspective they mention sourcing, delivery, support, monitoring, and control as key activities of IT governance [2].

Weill (2004) defines IT governance "[…] as specifying the framework for decision rights and accountabilities to encourage desirable behavior in the use of IT" [22]. He further shows the different fields of IT decision needs and presents different IT governance archetypes. The five most important IT decision needs are in the areas of IT principles, IT architecture, IT infrastructure strategies, business application needs, and IT investment. The IT governance archetypes are defined by "who makes each type of decision, who has input to a decision, and how these people are held accountable for their role". According to Weill, the different decision roles can be assigned to C-level executives, corporate IT, and business units or process owners [22]. EASs overlap with the competencies defined by both definitions of IT governance (e.g., sourcing of IT, distribution of IT, IT investment), so it will be important to discuss the consequences of EAS use on IT governance, and to propose potential strategies, i.e., to define the role of business and IT.

Organizational Buying Behavior and Enterprise Software Acquisitions. From an Industrial Marketing point of view, software procurement is an instance of organizational buying and an EAS can be defined as "a set of organizational and technological means constituting a centralized infrastructure serving a (individual or organizational) software consumer throughout the buying process" [23]. Robinson et al. (1967) developed a framework to identify organizational buying situations and introduced three "buying classes": new task, modified rebuy, and straight rebuy [24]. According to Webster and Wind (1972), the buying process is carried out by a buying center – the set of all the individuals from the buying organization taking on a role in the decision process (typical roles are: influencer, decider, user) [25] and involve different organizational units, such as the IT department, business units, the purchasing department, or workers council. The vendor in turn compiles a "selling center" [26] to approach the different interests of the customer stakeholders.

Based on the early works in organizational buying behavior, researchers have investigated factors of influence in the organizational buying process: for example, Sheth (1973, 1996) distinguished individual, environmental, and group-organizational aspects [27, 28]. Few authors have researched organizational buying in the context of software purchases. Based on Webster and Wind (1972), Halingten and Verville (2002; 2003) studied the purchase of ERP systems and classified influencing factors grouped into environmental, organizational, interpersonal, and individual factors. In addition, they analyzed the acquisition process and defined the phases planning, information search, selection, evaluation, choice and negotiation [4, 29].

Technology Adoption and Acceptance. An enterprise app store itself can be seen as an IT innovation. Hence, technology adoption and acceptance is an important field of research to investigate the use of EASs by companies and end users. The Technology–Organization–Environment framework (TOE) states that innovation adoption decisions by organizations are influenced by the technological, organizational, and environmental context [30]. The theory most widely used in IS research to study adoption decisions by individuals is the Technology Acceptance Model (TAM) [31]. It proposes mainly two independent variables influencing the individual's intention to use a specific technology: perceived usefulness and perceived ease of use.

Software Platforms and Ecosystems. Multiple publications in software platform and ecosystem research mention that an "online marketplace" is at the heart of software platform offerings (e.g., Platform-as-a-Service, PaaS) [32–34]. PaaS as an independent market offering constitutes hardware, software, and service components in order to enable independent software vendors (ISVs) to develop and to provide software solutions to customers [32]. The marketplace or the "platform store" is mainly evaluated in its role of an intermediary (i.e., cybermediary) to market the software products developed by ISVs on top of the software platform to customers [34]. Giessmann et al. define such a marketplace as follows: "The PaaS provider maintains a marketplace where customers can buy software components. The marketplace can offer provisions for software requests […]" [33]. The PaaS provider is therefore offering both the software platform to develop software components and the online marketplace to market and distribute them. Moreover, the PaaS provider often offers its own software components via the marketplace [32].

The understanding of a PaaS marketplace is largely equivalent to that of a public EAS. However, it is not necessarily required for the public EAS provider to also offer a software development platform to ISVs.

Enterprise App Stores. The EAS model as such has been the subject of only a few scholars so far, though business and technology analysts (e.g., Gartner, IDC) regularly rank it among the top strategic IT trends [35–37].

Novelli and Wenzel (2013) have qualitatively researched the app store model for enterprise software (i.e., public EAS) and identified drivers and barriers with regards to the organizational adoption of an EAS for different types of enterprise software (core solutions, on-top solutions, usage enhancements, IT services) [17]. Moreover, they have coined the term "app'ification" in the context of enterprise software referring to "app'ified" software if an application is suited for online sales and distribution due to its characteristics such as focused scope, trial available, starter package, or instant deployment. Furthermore, they provided recommendations on how to best integrate EASs with traditional, "offline", direct sales channels [23].

Beimborn and Palitza (2013) have investigated the benefits of internal EASs [13]: they define an internal EAS as a software system to provide functions of MAM. MAM complements Mobile Device Management (MDM) in corporations by managing the lifecycle of mobile apps, including development, procurement, distribution, configuration, update, and removal. They have identified benefits of using an internal EAS in the areas of IT compliance, app lifecycle management, and

total cost of ownership (TCO). Internal EASs as described in their study are originating from the mobile app segment; however, they are not restricted to this domain and can be applied to other types of software applications as well.

Neither of the works elicited the different use cases and capabilities of public and internal EASs in detail, nor did they contrast the two models.

3 Methodology and Research Process

In this study we have opted for an explorative, qualitative research strategy and have chosen an ideographic and comparative case study design [18, 38]. Exploratory research is preferred in novel contexts where only limited empirical data is available or the structures and themes of the studied phenomenon are still unclear. A case study design is used to investigate the unique characteristics of one or multiple cases (i.e., ideographic) in contrast to cross-sectional designs, where the focus lies on nomothetic, generic findings. Comparative designs embody the logic of comparison: they intend to better explain a phenomenon when two or more cases are selected and contrasted or put in relation [18]. EASs are a novel socio-technological context and come in two distinct models. By intensively studying internal and public EASs, we try to reveal the most important features, processes, and consequences for software customers. Further, the study design allows us to compare the two models and to derive additional knowledge which would not have been possible in two unrelated studies of the two EAS models. As organizational case study context SAP has been chosen. SAP fulfilled multiple criteria at the same time: SAP is one of the largest vendors of enterprise software; it operates a public EAS, the SAP Store [19], and offers an internal EAS [20]; it provides software platforms and related PaaS offerings to enable ISVs to develop and market their own applications [39]; and has a comprehensive MDM and MAM portfolio to manage enterprise mobility within companies [40]. Fig. 1 illustrates the sequential research process of this study.

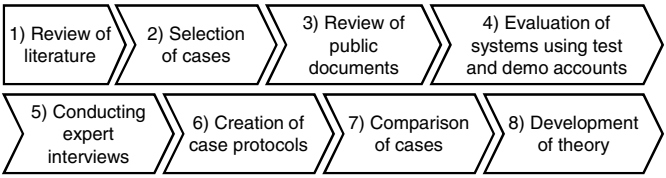

Fig. 1. Simplified, sequential research process

After a literature review and case selection, publicly available material was reviewed (online documentation: the SAP Store Support Guide [41] and the SAP Enterprise Store help documentation [20]). Subsequently, we have been provided with a test account to the SAP Store [19] and a demo system of the SAP Enterprise Store to analyze the functionality of the respective systems. Lastly, two experts from the product management of the SAP Store and the SAP Enterprise Store were interviewed. During the interviews we mainly tried to confirm assumptions and discuss open questions which could not fully answered by the public material and or

test systems. The interviews lasted 45 minutes each and notes were taken during the interviews. All findings (public material, test systems, interviews) were recorded and integrated in case study protocols. They formed the basis for the comparison of the two cases and were used to derive implications.

The public and internal EAS where studied along various categories: First, their respective value system according to Porter (1985) [42] is analyzed to identify all relevant stakeholders their roles and relationships. Second, the catalogs are reviewed, i.e., which applications are supported by the respective EAS. Last, a detailed functional analysis is conducted. Though, we tried to keep the categories in the functional analysis equal, they partially differ. This is due to the fact, that the internal and public EAS essentially are two different models and support partially different processes. To compare public and internal EAS, we used the IT governance process [2] to structure the comparison and assess the individual impact.

4 Presentation of Cases

In this section the two cases SAP Store and SAP Enterprise Store are presented (cf. Fig. 2). First, the value system is reviewed. Second, the supported applications (Catalog Management) are analyzed and third, a functional analysis is performed.

Fig. 2. Screenshot SAP Store (left) and SAP Enterprise Store (right, demo system)

4.1 Case: SAP Store

Value System. There are three main roles (cf. Fig. 3) in the value system of the SAP Store. The software customer uses the SAP Store as an online, self-service environment to conduct software acquisitions. ISVs provide the applications on the SAP Store and use it as a global, online sales and distribution channel.

The ISVs are members of the SAP Application Developer Partner program [39]. SAP is operating the SAP Store and fulfills the tasks of an intermediary and broker (e.g., providing app store infrastructure, catalog maintenance, certification and quality assurance, delivery). SAP collects a revenue share from the ISVs and thus participates in each sales transaction. Further, SAP provides its own solutions via the SAP Store as well and uses it as an online sales channel.

Catalog Management / Offering. At the time this study was conducted (August to November 2013), the SAP Store offered approximately 1200 solutions. This number differs slightly depending on the selected country. About half of the solutions are provided by SAP and the other half are provided by around 400 SAP partners. The solutions span all industries, business areas, deployment models (on-premise, cloud, mobile devices, desktop), and range from simple to highly complex applications.

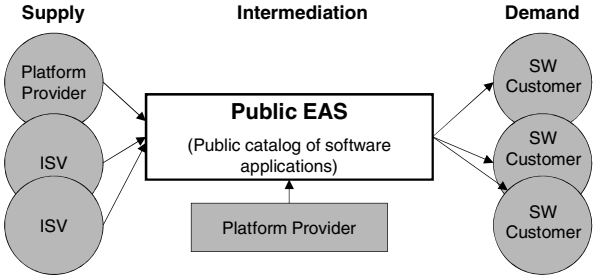

Fig. 3. Simplified value system of the SAP Store

Functional Analysis. The SAP Store supports the entire buying process from information search, evaluation, negotiating and buying, and delivery (derived from Verville and Halingten). Below, the functionality will be evaluated for each process step separately complemented with non-process-related functional categories.

Information Search. The SAP Store catalog is organized by multiple categories: industry, line of business, or categories for special topics, such as "Analytics" or "Cloud Solutions". In addition to browsing, filtering, and searching the catalog, there is a recommender providing personalized recommendations to the user. Three main screens support the phase of identifying a solution and gathering information. The SAP Store homepage highlights selected solutions and gives personal recommendations to the user. The catalog view is a typical list view: it shows search results and provides filters. Key information such as short descriptions or price are visible in this view and the key transactions can be accessed (e.g., add to cart). The solution view provides detailed information on a solution, e.g., functional, technical, and pricing details, introduction videos, customer success stories, and screenshots.

Evaluation. The SAP Store provides several dedicated features to evaluate solutions in more detail. Most solutions come with a trial version or demo mode. Simple solutions (e.g., which can be installed on the desktop) or which are operated in the cloud often have trials with most functionality enabled and integrated sample data. These trials are often limited to a certain period (e.g., 30 days). Other solutions come in a free version and a paid version. The free version can be downloaded and used without time limitations and can serve for evaluation or simple use cases.

Mobile enterprise apps can be downloaded for free (only the backend components need to be purchased) and include a demo mode to try the app. System landscape requirements can be evaluated with the so-called "compatibility check". This feature

analyzes the customer's system landscape and assesses prerequisites of the selected solution. The compatibility check informs the user if additional components need to be acquired and installed first. The SAP Store provides a selection of social features which may also be used to better evaluate a solution. Users can write reviews for purchased solutions and provide ratings which are visible to other customers. Further, users can recommend a solution to colleagues by using established social networks such as LinkedIn or simply e-mail.

Negotiation and Purchase. If the user decides to purchase a solution, he can add it to the shopping cart. The shopping cart shows detailed price information (e.g., fixed and recurring fees) and the quantity of items can be changed in this screen. As payment option, the user can choose between invoice and credit card. The SAP Store also recognizes corporate customer or volume discounts. Once the user proceeds, the review screen is shown: in this screen the terms and conditions of the selected solutions need to be reviewed and agreed to. Before submitting the order, the user can also enter an internal "Purchase Order ID" for correct booking of the order with his company's purchasing system.

Delivery. The delivery of the solution depends on the deployment model. On-premise server solutions, mobile apps, or desktop solutions are delivered via direct file downloads (download link is sent via e-mail to the buyer); cloud solutions are simply activated and the buyer receives a link with login instructions. The invoice is sent to the buyer in a separate e-mail.

Users and Permissions. There are multiple predefined user roles in the SAP Store with different permissions. A "Guest User" is an unregistered user on the SAP Store. He can freely browse the entire catalog and can even download free applications or demos. A "SAP Store User" is a registered user associated with a company. A SAP Store User can fully browse the catalog, use social features (review solutions, recommend solutions to colleagues), and buy a few selected solutions using his credit card (mainly personal solutions). The "SAP Store Buyer" has the additional permission to purchase all solutions available on the SAP Store on behalf of his associated company (invoice or credit card). Lastly, the SAP Store Super User can invite users to the SAP Store and pre-register them with his company. Further, he can assign and revoke SAP Store roles to the individual users.

Administration and Support. The "SAP Store Account" section allows users to access user management, review the details of past SAP Store orders, and view and accept quotes. A dedicated support page helps users with questions. Further, SAP provides phone numbers, an e-mail address, and a live chat if users have questions with regards to the SAP Store or a desired solution.

Complex Buying Scenarios. The SAP Store is a self-service buying and e-commerce platform. However there are dedicated features to use the SAP Store in combination with a traditional offline, direct sales channel, e.g., a SAP sales representative. Most solutions on the SAP Store also provide a "Contact Me" button to ask for the involvement of a salesperson or to provide a "request a quote" transaction. A

salesperson then creates this quote offline and loads it back up to the SAP Store. There, customer users can review and accept (or decline) the quote.

4.2 Case: SAP Enterprise Store

Value System. The SAP Enterprise Store is an internal EAS. It is available for purchase as a single product or can be acquired as part of the SAP Mobile Secure suite (SAP, 2013). The Value System can be described using four roles. The SAP Enterprise Store is hosted by SAP and provided to the customer as a cloud solution. The customer's IT department is managing the behavior and fully determines which applications or digital content is available. Employees can discover applications and content, and can download, activate, or consume them. Furthermore, the SAP Enterprise Store can be used to position selected content not only to company-internal consumers, but also to its ecosystem, such as suppliers, partners, subsidiaries, or end customers. This way it can be used to enhance the reach of corporate IT beyond the borders of the company (cf. Fig. 4). The SAP Enterprise Store can be integrated with the SAP Store to pre-fill the catalog with selected solutions from the public EAS. Further, it integrates with SAP Afaria, an MDM solution. In this case the SAP Enterprise Store replaces the App Catalog as part of Afaria, which is a rudimentary version of an internal EAS.

Catalog Management / Offering. The catalog is fully managed by the customer's IT department and can be used to host different kinds of digital content: mobile apps, desktop apps, web apps, or e-learnings. The applications and content provided target individuals, i.e., end users. Hence, the SAP Enterprise Store is not meant to distribute an entire server application, but it would in this case provide the user-related interface, which could be an app or even only a link and license activation.

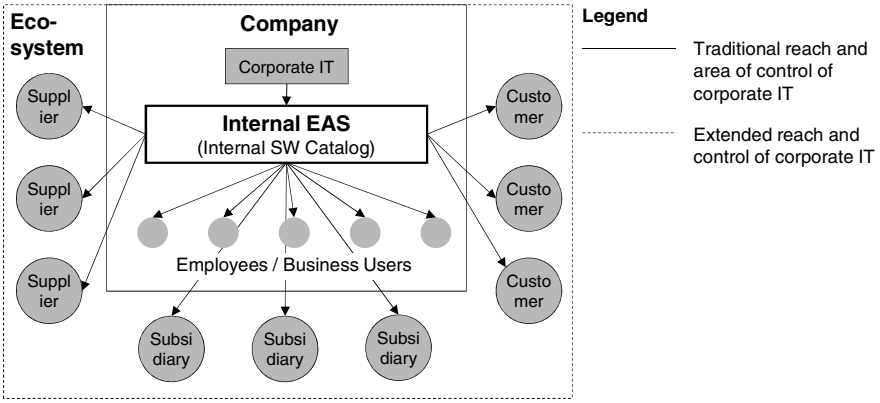

Fig. 4. Simplified Value System of the SAP Enterprise Store

The SAP Enterprise Store can host solutions which are available in the SAP Store, applications developed in-house, or any third-party application. The categories to browse the catalog can also be freely configured by the administrator.

Functional Analysis. The functional analysis is presented along slightly different categories than the SAP Store as the focus lies on the adoption of single applications by individual users and the management of these applications by an IT department.

Information Search. The SAP Enterprise Store is similarly organized as the SAP Store and has a comparable look and feel. It can be accessed via the browser or apps available for mobile devices. On the homepage the user can enter the catalog by categories (defined by the company), search for applications, or select recommendations. In the list view, the user can filter the results using multiple criteria (e.g., device, line of business). In comparison to the SAP Store the list view shows more details with regards to device compatibility (e.g., phone, tablet). The solution view displays details of the solution (screenshots, videos, functionality, and technical details). In contrast to the SAP Store, there are no dedicated pricing or commercial details but more details on device compatibility. The available user reviews can further be filtered with regards to a device-specific version. The download or deployment of an app can be triggered in the list view and the solution view.

Evaluation, Negotiation, and Purchase. To evaluate a solution, the user can view screenshots, videos, or ratings and reviews. Dedicated features to access trials or demos are not available. Once a user decides to use a solution he can download it or trigger the delivery. A shopping cart or dedicated buying process is not supported.

Delivery. The delivery depends on the deployment model of the app and on which device the SAP Enterprise Store is accessed. For mobile apps the installation of an app is directly triggered if the catalog is accessed from the mobile device itself. Where a mobile app is chosen from the desktop version of the catalog, an e-mail including the installation link is sent to the user's e-mail address. For desktop apps or other digital content (e.g., e-learnings) a file download is triggered. For web apps login credentials and activation links are sent to the user's e-mail address. The SAP Enterprise Store can also be integrated with SAP Afaria, the MDM solution of SAP. In this case, a selected app is handed over to the MDM system for device deployment.

Company-Internal IT Innovation Process and Social Features. The SAP Enterprise Store cannot only be used to distribute productive applications but also allows in-house developers to upload their own apps, even if these apps are still in development. This is supported by a dedicated process and status management: if an in-house developer uploads an application he can mark it as "In Development", "Beta", or "Productive". Before the app is published, the administrator needs to review and approve it first. Eligible users can then review and try these non-productive apps and provide feedback via the social features of the SAP Enterprise Store. In general there are features to review and rate applications and this way give feedback to the IT department.

Users and Permissions. The SAP Enterprise Store integrates with corporate identity systems and recognizes users in the intranet. Out of the box there are only two roles: a standard user who can fully browse and download apps and the administrator role. The administrator can manage the catalog, view usage statistics, adjust the visual appearance, create catalog categories, assign users to roles, or even restrict certain

catalog content to a group of users or roles. The standard roles can be enhanced with additional roles such as developers (eligible to upload apps), beta users (employees who have access to non-productive apps), or dedicated roles for suppliers or customer companies who only have access to a sub-set of applications.

Administration. The administration of the SAP Enterprise Store is grouped into two categories: Setup and Statistics. Setup includes general settings, visual style settings (e.g., apply corporate branding), catalog categories, app management (publish, edit, retire, approve/decline new apps), and user and role management. The Statistics view shows detailed usage information of the SAP Enterprise Store such as number of downloads or uploads and can also be used for license management.

5 Comparison of the Public versus Internal EAS

In this section we will discuss how the two presented EAS models help to resolve the initially stated dilemma: how to involve business units and users in the IT governance and how to establish a bottom-up IT adoption process to satisfy the needs of today's tech-affine employees, and at the same time how to keep control of IT processes to

Table 1. Comparison of key capabilities of the public versus internal EAS model with focus on business involvement and IT control along the IT governance process (cf. [2])

IT Gov. Process	SAP Store (public EAS)	SAP Enterprise Store (internal EAS)
IT Sourcing	▪ Business can identify, gather information about, and try new business applications ▪ IT defines buyers and proactively invites business reps to participate during external sourcing ▪ IT can enable selected business reps to make purchases	▪ Early involvement of business users in in-house development projects (internal sourcing)
IT Delivery	▪ Instant delivery of software can accelerate delivery process	▪ Business users select and consume apps in a self-service mode using a consumer-friendly app catalog ▪ Provide apps to ecosystem ▪ Secure and instant delivery to user devices
IT Support		▪ Internal EAS can be used to distribute updates of applications ▪ Distribution of e-learnings
Monitoring	▪ IT can monitor all purchases on the EAS via a central order view	▪ Monitoring of app usage, downloads, reviews, ratings ▪ License monitoring
IT Control	▪ Define buyer roles ▪ Prevent business users from buying non-authorized app-lications	▪ Define target groups for applications (who can access which apps) ▪ Fully define catalog content and visual styling of EAS

meet corporate IT targets. In general, it must be said that neither EAS model predefines who in the organization takes over which decision role in the areas of IT governance (cf. Weill, [22]). Companies still need to define themselves how EASs are implemented and used in the organization, i.e., who takes over which part in decision processes. Table 1 presents the capabilities of the two EAS models with regards to business involvement and IT control in the main tasks of IT governance (cf. [2]).

In summary, the studied public EAS focuses on external sourcing, i.e., acquisition, of new enterprise software. It can be used to tightly involve business users during the identification and evaluation of applications. Dedicated capabilities are provided to enable organizational buying (e.g., buyer roles, corporate discount, quotations, compatibility check). Still, the provided features allow IT departments to control which applications are purchased and buying permissions can be managed actively. The self-service paradigm of the public EAS has the potential to increase efficiency during external sourcing activities and thus shorten buying cycles, and ultimately accelerate the introduction of new enterprise software applications.

The internal EAS model focuses on the distribution of personal applications to employees and the ecosystem. Thereby, it encourages an employee- or rather user-driven pull model in favor of a push model. IT departments can fully control the available content in the EAS catalog. The breadth of supported types of content (web, mobile and desktop apps, and other digital content) allows the internal EAS to be the single source for employees' app needs. The roles concept can further be used to target specific user groups: to pre-filter the available applications and even to position selected applications to the ecosystem. Another noteworthy capability is to actively support the internal innovation process, for example by publishing beta apps to a selected group of users to receive early feedback. Lastly, the monitoring capabilities of the internal EAS inform the IT department about which applications are used and which not and help to establish a more accurate and compliant license management.

Whether the EAS models will be accepted by business users can be analyzed in terms of "perceived usefulness and ease of use" (cf. [31]) . Perceived usefulness will be rated first and foremost by the applications provided by the EAS. Whereas the public EAS catalog is controlled by an external provider, the internal EAS catalog is filled by the IT department. In both cases it is important to offer a broad assortment of attractive applications. The second determinant "perceived ease of use" will be rated according to how convenient software acquisition or adoption processes can be conducted using an EAS. Both studied EASs are using a design very similar to consumer app stores or well-known e-commerce sites. Hence, it is safe to state that most users will be familiar with the interaction patterns of the researched EASs.

Combination of Public and Internal Enterprise App Stores. Our case study has shown that both models support the involvement of business users during software selection and adoption, though they focus on different parts of the overarching process. Fig. 5 shows a potential combined use of public and internal EASs for a given company along the software adoption process.

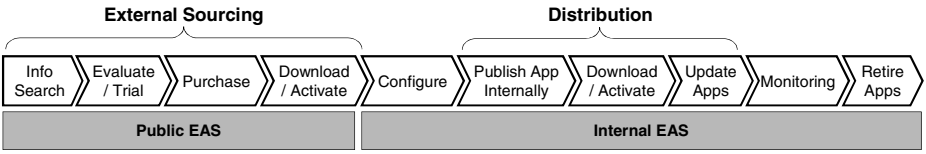

Fig. 5. Combined use of public and internal EASs

The public EAS can be used for external sourcing of new enterprise software and for actively involving selected business representatives during the acquisition process. After configuration of the acquired software for corporate use and defining the target user groups, the internal EAS provides employees with a consumer-like experience to discover and consume (i.e., download/activate) corporate applications. Further, the internal EAS helps IT departments to manage and monitor applications along their firm-internal lifecycle.

6 Limitations and Conclusion

In this study we used an explorative, qualitative research strategy and a case study design. Generalizability of qualitative research in general, but especially of case studies, is low [18]. Further, we did not involve experts from customer companies as we could not recruit adequate candidates at the time of this study (the SAP Enterprise Store was released in late summer 2013). Future studies (qualitative or quantitative) may research EAS models in the actual customer context and analyze other EASs available on the market.

We studied public and internal EAS models by researching two cases, the SAP Store and SAP Enterprise Store. The capabilities of each EAS were analyzed in detail and scenarios for practical use were discussed. Special focus was placed on how the two EAS models help to involve business users during key IT governance processes, especially sourcing and delivery of IT, while keeping control of IT. Moreover, we proposed a combined use of public and internal EASs and outlined options how these can be used in organizations. In this context we encourage future research to further study combinations of public and internal EAS especially in the context of real-world scenarios. As a concluding remark, we believe EASs will spur the digitization of the software acquisition and distribution process and will enable faster adoptions of IT innovations and a stronger involvement of business stakeholders.

References

1. Wenzel, S., Novelli, F., Burkard, C.: Evaluating the App-Store Model for Enterprise Application Software and Related Services. In: Wirtschaftsinformatik Proceedings 2013 (2013)
2. Meyer, M., Zarnekow, R., Kolbe, L.M.: IT-Governance - Begriff, Status quo und Bedeutung. Bus. Inf. Syst. Eng. 45, 445–448 (2003)

3. Liao, X., Li, Y., Lu, B.: A model for selecting an ERP system based on linguistic information processing. Inf. Syst. 32, 1005–1017 (2007)
4. Halingten, A., Verville, J.C.: A qualitative study of the influencing factors on the decision process for acquiring ERP software. Qual. Mark. Res. An Int. J. 5, 188–198 (2002)
5. Edison, H., Bin Ali, N., Torkar, R.: Towards innovation measurement in the software industry. J. Syst. Softw. 86, 1390–1407 (2013)
6. Lamendola, M.: Justifying Software Purchases, http://ecmweb.com/content/justifying-software-purchases
7. Gunasekaran, A., Love, P.E.D., Rahimi, F., Miele, R.: A model for investment justification in information technology projects. Int. J. Inf. Manage. 21, 349–364 (2001)
8. Weiß, F., Leimeister, J.M.: Consumerization - IT Innovations from the Consumer Market as a Challenge for Corporate IT. Bus. Inf. Syst. Eng. 4, 363–366 (2012)
9. Niehaves, B., Köffer, S., Ortbach, K.: The Effect of Private IT Use on Work Performance-Towards an IT Consumerization Theory. In: Wirtschaftinformatik Proc. 2013, pp. 39–53 (2013)
10. Carr, N.: IT doesn't matter. Harv. Bus. Rev. (2003)
11. Vizard, M.: CIOs Struggle With Relevance of Role to Business, http://www.cioinsight.com/it-management/cios-struggle-with-relevance-of-role-to-business
12. Jones, D., Behrens, S., Jamieson, K., Tansley, E.: The Rise and Fall of a Shadow System: Lessons for Enterprise System Implementation. In: ACIS 2004 Proceedings (2004)
13. Beimborn, D., Palitza, M.: Enterprise App Stores for Mobile Applications - Development of a Benefits Framework. AMCIS 2013 Proceedings (2013)
14. SAP: Simple UI for SAP Applications: SAP Fiori Improves User Experience, http://en.sap.info/sap-fiori-improves-user-experience/98278
15. Wenzel, S., Faisst, W., Burkard, C., Buxmann, P.: New Sales and Buying Models in the Internet. In: MKWI 2012, pp. 639–651 (2012)
16. Böckle, R.: B2B App Stores - Anbieter im Vergleich. Computerwoche 49, 14–21 (2013)
17. Novelli, F., Wenzel, S.: Adoption of an Online Sales Channel and "Appification" in the Enterprise Application Software Market. In: ECIS 2013 Proceedings (2013)
18. Bryman, A., Bell, E.: Business Research Methods. Oxford University Press, USA (2011)
19. SAP: SAP Store: http://www.sapstore.com
20. SAP: SAP Help - SAP Enterprise Store, http://help.sap.com/ses
21. Harris, J., Ives, B., Junglas, I.: IT consumerization: when gadgets turn into enterprise IT tools. MIS Q. Exec. 11, 99–112 (2012)
22. Weill, P.: Don't just lead, govern: How top-performing firms govern IT. MIS Q. Exec. 3, 1–17 (2004)
23. Novelli, F., Wenzel, S.: Online and Offline Sales Channels for Enterprise Software: Cannibalization or Complementarity? In: ICIS 2013 Proceedings (2013)
24. Robinson, P.J., Faris, C.W., Wind, Y.: Industrial Buying and Creative Marketing (1967)
25. Webster, F.E., Wind, Y.: A General Model for Understanding Organizational Buying Behavior. J. Mark. 36, 12–19 (1972)
26. Puri, S.J.: Industrial Vendors' Selling Center: Implications for Sales Management. J. Bus. Ind. Mark. 7, 59–69 (1992)
27. Sheth, J.N.: A Model of Industrial Buyer Behavior. J. Mark. 37, 50–56 (1973)
28. Sheth, J.N.: Organizational buying behavior: past performance and future expectations. J. Bus. Ind. Mark. 11, 7–24 (1996)
29. Verville, J., Halingten, A.: A six-stage model of the buying process for ERP software. Ind. Mark. Manag. 32, 585–594 (2003)

30. Tornatzky, L.G., Fleischer, M., Chakrabarti, A.K.: The processes of technological innovation (1990)
31. Davis, F.D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Q. 13, 319–340 (1989)
32. Beimborn, D., Miletzki, T., Wenzel, S.: Platform as a Service (PaaS). Bus. Inf. Syst. Eng. 3, 381–384 (2011)
33. Giessmann, A., Stanoevska, K.: Platform as a Service–A Conjoint Study on Consumers' Preferences, pp. 1–20 (2012)
34. Scholten, S.: Platform-based innovation management: a framework to manage open innovation in two-sided platform businesses. Universität Stuttgart, Stuttgart (2011)
35. Pettey, C.: Gartner Identifies the Top 10 Strategic Technology Trends for 2013 (2013), http://www.gartner.com/newsroom/id/2209615
36. Pettey, C.: Gartner Identifies the Top 10 Strategic Technologies for 2012 (2012), http://www.gartner.com/newsroom/id/1826214
37. Drake, S.D.: Mobile Enterprise App Stores: Open for Business. IDC (2012)
38. Darke, P., Shanks, G., Broadbent, M.: Successfully completing case study research: combining rigour, relevance and pragmatism. Inf. Syst. J. 8, 273–289 (1998)
39. SAP: SAP Application Development Partner Center, http://www.sapadpc.com
40. SAP: SAP.COM - Secure Apps, http://www.sap.com/pc/tech/mobile/software/solutions/device-management/secure-apps.html
41. SAP: SAP Store Support, https://store.sap.com/sap/cp/ui/resources/store/html/CustomerSupport.html
42. Porter, M.E.: Competitive Advantage: Creating and Sustaining Superior Performance. The Free Press, New York (1985)
43. Salesforce.com: AppExchange, https://appexchange.salesforce.com/
44. Microsoft: Microsoft Pinpoint, http://pinpoint.microsoft.com/
45. Amazon.com: Amazon AWS Marketplace, https://aws.amazon.com/marketplace
46. Google: Google Apps Marketplace, https://www.google.com/enterprise/marketplace/
47. Symantec: Symantec App Center, http://www.symantec.com/mobility/products
48. App47.com: App47, http://www.app47.com/
49. Salesforce.com: Private AppExchange, https://appexchange.salesforce.com/listingDetail?listingId=a0N3000000B4VlZEAV
50. OpenPeak: OpenShop, http://www.openpeak.com/OpenShop.html

Impact of Cloud Computing Technologies on Pricing Models of Software Firms – Insights from Finland

Gabriella Laatikainen and Eetu Luoma

Department of Computer Science and Information Systems,
University of Jyväskylä,
Jyväskylä, Finland
{gabriella.laatikainen,eetu.luoma}@jyu.fi

Abstract. In this paper we study the changes in the pricing models of software firms that use cloud computing technologies as part of their products and services. This paper presents findings from 324 responses to a questionnaire survey on how pricing model elements of software firms have changed as a result of adopting hardware virtualization, multi-tenancy, online delivery and configurability. The findings suggest that Software-as-a-Service firms – making use of the cloud computing technologies – are generally simplifying their pricing model, increasing the use of usage-based pricing, reducing the customers' influence and unifying their pricing across customers. These changes occur together with standardization of their products or services. The findings provide a view to the transformation of the software industry, characterized by both technological and business model redesigns.

Keywords: cloud, SaaS, pricing, software firms, business models.

1 Introduction

Software-as-a-Service (SaaS) is both a delivery and a business model for software firms defined by technological and business characteristics. Recent literature describes SaaS as the delivery of multi-tenant, virtual, web-based and configurable application that is accessible through browser [1]–[4]. Applying these technological characteristics to its application enables a software firm to offer a cloud computing service with the essential cloud characteristics to its customers. Viewing SaaS from the business perspective, the model is understood as offered through a different revenue logic compared to the traditional licensed software, such as subscription-based and/or usage based pricing [2], [5]–[7].

Introducing cloud technologies therefore implies changes not only to software architecture but also to business model design. Among the business model elements, a well-designed revenue logic is a key condition for commercial success. Pricing models influence not only the demand, but have an effect also on the way how users use the product or service, and have a long-term influence on customer relationships [8]. The revenue logic can also differentiate a product from the competitors and this way increase the company's revenues [9]. However, even though pricing is a powerful

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 243–257, 2014.

strategic tool in manager's hands, it also causes challenges to software firms that develop SaaS to the market. Information is often difficult to price and the currently observed constantly changing labyrinth around software pricing makes pricing even more complex [3], [10]–[13].

With the emergence of cloud technologies, the software market evolves rapidly and the firms' needs for strategic changes increase. Different studies in current literature focus on software firms' revenue logic and their products and services. However, despite of its importance, there is a shortage of empirical evidence on how the software firms *changed* their pricing models due to adopting cloud computing technologies. This study fills the gap by analyzing 324 Finnish software firms to find out (1) what are the changes in pricing model that are caused by cloud computing technologies, such as virtualization, multi-tenancy, online delivery and configurability; and (2) whether changes in pricing model elements are caused directly by cloud computing technologies or through changes in the firms' products or services offered to their customers.

The contribution of this paper is two-fold. First, researchers gain a better understanding on how the cloud technologies transform the software industry and how firms change their value proposition and pricing model after adopting cloud technologies. Secondly, the managerial implications provide insights into how particular cloud technologies affect different aspects of pricing.

The structure of this article is as follows. In the next section, we give an overview on recent work related to value proposition and revenue logic as key business model elements in the context of cloud technologies and describe the hypotheses of this research. In Section 3, we describe the research methodology used in this article. In Section 4 we present the findings of our analysis. We conclude our paper with discussion and summary in sections 5 and 6, respectively.

2 Theoretical Background

2.1 Business Models

Business model is a conceptual model of a business: a description of how a company organizes itself, operates and creates value [10], [14]–[17]. The static view on business models sees them as a blueprint for the coherence between core business model components [18]. Besides others, the core business parameters include value proposition incorporating the product/service portfolio [14], [15], [18], [19] and revenue logic referring to the structure of income [14], [15], [19].

On the other hand, the dynamic view uses the business model concept as a tool to address change and innovation in the firm or in the model itself [18]. Changes in the model itself can be related to the different phases of the lifecycle of business models, such as creation, extension, revision and termination [20]. The reason for these changes might be a response to external and/or internal influences. In the literature, the advances in contemporary technology are argued to be a key external factor that leads to changes in business strategies and processes [17], [19], [21]–[24]. Moreover, Chesbrough and Rosenbloom [19] argue that the financial performance of a given

firm is associated with developments in firm's environment, but *only* through changes in the firm's business model. Besides the external influences, the need for business model changes might also come internally. Business models are designed, implemented and changed by employees of the company who make decisions based on their perception of the firm's environment [18]–[20]. As a consequence, the elements of business models are interrelated and changes in one of the components might cause changes also in others [18].

There is currently little in the literature that empirically examines just how exactly software providers do convert to supplying SaaS. A couple of exceptions to this are the studies by Stuckenberg et al. [6], Ojala and Tyrväinen [25] and Novelli [26]. While their findings are based on rare cases, they both seem indicate a trend towards offering more standardized products and services, increasing customer-facing activities and changes in revenue logic towards subscription-based pricing.

2.2 Value Proposition and Cloud Technologies

As a core item of business model, value proposition communicates the value that the companies' product/service portfolio creates for the target customers using technology [19]. In software industry, the product/service portfolio incorporates the set of functionalities of the software, the needed infrastructure and the deployment, delivery and maintenance of the software [27], [28]. Specifically, software firms that develop SaaS to the market companies employ cloud technologies in their value proposition, such as hardware virtualization, multi-tenancy, and web service [1]. Besides, a cloud mature application should also be configurable [2]. These four technologies give the software firms the means to introduce SaaS service to the market, a service which has the essential cloud computing characteristics of on-demand self-service (through configurability), network access (web service), resource pooling (virtualization and multitenancy) and elasticity (virtualization), as they're described in the reference definition of cloud services [29].

Hardware virtualization offers an abstract computing platform to the users instead of the physical characteristics, such as raw computing, storage, network resources [1]. Virtualization also enables encapsulation for the applications, so that they can be installed, configured and maintained [30].

In a multitenant architecture, a single instance of common code and data is shared between multiple tenants [31]. Besides the requirements of shared hardware resources, shared application and shared database instance, Bezemer et al. [32] requires also high degree of configurability in look-and-feel and workflow from multitenant software. Some researchers consider also multi-instancy as a form of multi-tenancy [33], where a vendor hosts separate instances for each customer within shared hardware [33], [34].

Web service represents communication over the HTTP protocol, where the customers use a browser to use the application [1]. SaaS is therefore also a delivery model, software that is available through the network.

Configurable software offers the possibility for users to modify the application's appearance and behavior through metadata services to meet their needs. These

configuration changes might refer to user interface and branding (graphics, colors, fonts, logos, etc.), workflow and business processes, extensions to the data model and access control [2].

2.3 Revenue Logic and Software Pricing Models

The revenue logic describes the structure of revenues, how the company makes money by serving its customers [14], [18]. In software industry, the most common revenue streams are: i) monthly or annual subscription fees, ii) advertising based revenue, iii) transaction based revenue (customers are charged based on the number of transactions they perform), iv) premium based revenue (revenue is generated from charging for premium versions besides the free versions), v) revenue from implementation and maintenance services and vi) software licensing [28], [35]–[37].

Software pricing in these above introduced revenue models may base on different aspects. The software pricing model parameters of Lehmann and Buxmann [38] and the SBIFT model of Iveroth et al. [39] are taken into account in the classification of cloud pricing models that describes these models along 7 dimensions [40]:

1. **Scope** represents the granularity of the offer, whether it is priced as a package or different prices are given for different functionalities.
2. **Base** represents the information base the price is set on. The price might be decided based on cost considerations, the competitors' prices, based on performance or customer value.
3. **Influence** represents the ability of buyers and sellers to influence the price, and it contains the options Pricelist, Negotiation, Result-based price, Pay-what-you-want, Auction and Exogenous pricing.
4. **Formula** represents the connection between price and volume, and it contains different variations of fix and variable price components.
5. **Temporal rights** represent the length of service's usage period, and it can be Perpetual, Subscription-based or Pay-per-use.
6. **Degree of discrimination** represents the level of price variety depending on the buyer. The software may be offered to the customers with a different price in different regions or with a price dependent on the time of buying. The price can depend on the acquired volume, software's quality, or it might be even customer-specific.
7. **Dynamic pricing strategy** represents the strategy of dynamic price change over time. Penetration, skimming or hybrid pricing strategies belong to this dimension.

It can be noted, that these 7 dimensions of cloud pricing framework are different by nature: the dimensions Base and Dynamic pricing strategy represents long-term, strategic decisions made usually by the upper management; while the other five dimensions describe the elements of pricing models that can be modified more easily.

We chose to use this framework as a starting point for the present study, since it provides the most state-of-the-art and the most integrative work in the current pricing literature in cloud context. The framework adopts general pricing model elements to

software business and cloud context, allowing researchers and practitioners to study different pricing model aspects in a systematic, holistic way.

SaaS business model has an altered licensing scheme compared to the traditional software business, where acquiring a perpetual use license represents the common method of transaction [5]. Instead, in SaaS the customer organization and the software firm agree on a subscription and the software firm develops, deploys and operates the software application in its datacenter of choice. This can been interpreted as separating the ownership of software from its use [41], [42], hence software is provided and consumed as a service rather than as a product. Contemporary SaaS pricing models have been studied notably by Lehmann and Buxmann [38], [43]. However, their studies focus on the current pricing models of SaaS vendors, rather than how pricing models have changed together with changes in technologies and value propositions.

2.4 Research Gap and Hypothesis Development for the Current Study

In our review of the extant literature, we searched for prior work related to cloud technologies, SaaS and pricing models but also the business model concept with a special focus on changes in business models that occurred as a response to technological changes. We found that different aspects of cloud computing have been received moderate attention from the researchers; however, despite of its importance, prior literature lacks empirical studies on how software firms *changed* their pricing models due to adopting cloud computing technologies. In current study therefore we focus on the role of cloud computing technologies in the pricing models of software firms.

In software business, as a result of technological changes and competitive forces, there is a gradual shift in business models towards increasing service revenues [28]. With the emergence of cloud computing, software firms not only implement technological changes by introducing multi-tenancy, hardware virtualization, configurability and internet-based delivery, but these technological characteristics imply also changes to the revenue logic. SaaS software is often offered through the subscription model billed monthly or even in shorter periods [38]. SaaS vendors may often provide their prices through pricelists on their websites [43], indicating more transparent and unified pricing across customers, where the influence of the customers on prices decreases [40]. A cloud solution is a result of co-operation of different value chain partners, where the SaaS provider might pass the usage-based pricing metrics derived from the PaaS provider to the end customers. Both customers and providers might prefer simple pricing models where different functionalities are bundled into one package with one price. [38], [40], [43]

Based on the claimed characteristics of software firms, we assess pricing model changes caused by introducing cloud computing technologies through changes in the pricing model elements and we hypothesize that:

H1. Adopting cloud computing technologies, i.e. introducing hardware virtualization, multi-tenancy, internet-based usage of the software and configuration through internet is associated with change towards 1) simpler pricing 2) less

negotiation 3) usage-based pricing 4) shorter contracts and 5) more unified pricing across customers.

SaaS software is argued to be more standardized than the traditional software: only a limited set of functionalities is provided to a larger market segment instead of customer-specific solutions [4]. Changes in value proposition imply changes also in other business model elements, such as the pricing model [10], [46]. Therefore, we assess whether pricing model changes are caused by changes in value proposition and we hypothesize that:

H2. Standardizing the value proposition, implementing a limited set of new functionalities is associated with change towards 1) simpler pricing 2) less negotiation 3) usage-based pricing 4) shorter contracts and 5) more unified pricing across customers.

3 Research Method

3.1 Data Collection

The goal of our empirical study was to capture changes in software firms' pricing models due to adoption of cloud technologies. The data used in this study was collected as part of the annual Finnish software industry survey whose primary aim is to gather the information about the current state of software industry. The definition of software firm followed the tradition of the Software Industry Survey[1], focusing on all Finnish companies whose main activity is to provide software as products or services to the customers. The details of the survey can be found online, so in this study we describe the sample and the data collection procedure only shortly.

The survey follows a modified version of the tailored design [44] and collects data using letters and web-based form with email invitations. The mailing list of the survey contained key informants of 4878 software firms. The data collection started in April and ended in June 2013. The respondents were contacted five times and the data gathering resulted in receiving 379 complete and 121 partial responses.

After collecting the data, we used a filter to select the companies appropriate for the goal of this study. As our focus was on firms providing Software-as-a-Service, which originate from either software product firms or software services firms, we excluded producers of embedded software and software resellers from the analysis. Further, since the objective of this study was to examine the factors causing changes in the firms' pricing models, we excluded software firms younger than two years from the analysis. In total, 324 usable responses from software companies matched our inclusion criteria and were used for the analysis.

3.2 Concepts and Their Operationalization

We conceptualized the pricing model of software firms through its dimensions in the cloud pricing framework [40]. The pricing model incorporates the granularity of the offer, the customers' negotiation level, the pricing formula consisting of fix and

[1] See `http://softwareindustrysurvey.org` for details about the survey.

variable price components, the temporal rights and price discrimination [38]–[40]. Cloud technology includes hardware virtualization, multi-tenancy, web-based software and configurability [1], [2]. Value proposition was conceptualized through the firms' product/service portfolio that is offered to the customers [19], [45].

Since the primary goal of the survey was different from the aims of this study, we had to choose between investigating specific changes in the pricing models with single-item measures or studying only one pricing aspect in detail. The aspects of SaaS pricing are diverse, therefore we could not follow the suggestion of the configuration approach [46] to measure one aspect and infer changes to the whole pricing model. Thus, we used single-item measurements for measuring and interpreting various pricing model changes.

The dependent variables of this study measure changes in software firm's pricing model during the last three years. We designed the variables based on the characteristics of assumed SaaS pricing models: capturing change toward having simpler pricing model (labelled "Scope"), toward less negotiation ("Influence"), toward usage-based pricing ("Formula"), toward committing to shorter contracts than before ("Temporal rights") and toward more unified pricing across the customers ("Discrimination"). We excluded the dimensions "Base" and "Dynamic pricing strategy" from our research setting due to their long-term, strategic nature and rather concentrated on different operative aspects on pricing models.

Measuring change in the value proposition was based on the assumption that SaaS firms standardize their products and services and implement fewer new functionalities to their products/services than before. The five dependent variables and the independent variable "Standardization" and "Fewer functionalities" were measured with the question "How well these statements describe the change of your company's business model during the last three years?", where response options were anchored ranging from "1=strongly disagree" to "5=strongly agree".

The independent variables measuring technology adoption are dummy (binary) variables that describe whether or not the companies use hardware virtualization (labelled "Virtualization") multi-tenancy (labelled "Multi-tenancy"), web-based software (labeled "Online delivery") and configurability (labelled "Configurability") in their products and services. These were measured by the question "Which cloud computing features were used in your company's products or services in 2012?", and had the options "Hardware virtualization", "Multi-tenancy", "Internet-based usage of product or service" and "Configuration through internet (Customer self-service)".

The control variables are the size and age of the company ("ln(Size)" and "ln(Age)", respectively). The proxy for the size of the company is the firm's revenue in 2012 and the company's age is determined based on the age of the firm in 2012. Using these control variables is justified. A larger company may have better resources to initiate and execute changes compared to smaller firms with limited resources. On the other hand, the more mature companies are likely to suffer from inertial forces within the organization that obstructs changes [47].

3.3 Data Analysis

In this study we used non-parametric correlations and multivariate ordinal regression analyses to investigate the hypotheses. In particular, non-parametric correlations are

used to reveal associations between cloud technologies, changes in value proposition and elements of pricing models. The ordinal regression analyses were employed to assess the pricing model changes attributable to adoption of cloud technologies and changes in value proposition. Ordinal regressions treat each ordinal value as an independent variable; thus it is possible to examine parameter estimates for a certain range of values within an independent variable [48].

Before running the data analyses, exploratory tests were carried out to choose the most appropriate statistical methods. Specifically, after realizing that the dependent variables were negatively skewed, we run the Shapiro-Wilk's test of normality and the test was significant. Thus, the sample did not come from normally distributed population; therefore we chose to use non-parametric statistics. We also investigated the potential presence of outliers. After exploring the data, we detected four influential responses visually using box plots and removed them from the analysis. Next, we applied Harman's single-factor test to check the common method variance problem, that is typical in case of survey research [49]. The unrotated factor solution did not reveal a single factor, which would account for the majority of the variance in the model, suggesting that the method variance would not be a problem in the data. Different concerns are related to the ordinal regression analyses, such as the multicollinearity of the independent variables, the choice of link function, and the proportional odds assumption. From the correlation statistics presented in the Table 1, we did not detect high correlations between the independent variables; thus, multicollinearity would not impede the results. Our choice of link function was driven by the distribution of the ordinal outcome as suggested by the literature [50], and we employed Cauchit for the

Table 1. Non-parametric correlations between the variables

	Spearman rho		1	2	3	4	5	6	7	8	9	10	11	12	13
1	Scope	Coefficient	1.000												
		Significance													
2	Influence	Coefficient	.222	1.000											
		Significance	.001	.											
3	Formula	Coefficient	.218	.106	1.000										
		Significance	.001	.104	.										
4	Temporal rights	Coefficient	.044	.045	.140	1.000									
		Significance	.501	.488	.032	.									
5	Discrimination	Coefficient	.489	.256	.185	.030	1.000								
		Significance	.000	.000	.005	.644	.								
6	Virtualization	Coefficient	.089	.042	.174	-.007	.141	1.000							
		Significance	.170	.519	.007	.911	.029	.							
7	Multi-tenancy	Coefficient	.171	.197	.230	-.093	.159		1.000						
		Significance	.008	.002	.000	.150	.014		.						
8	Online delivery	Coefficient	.131	.040	.182	-.025	.130			1.000					
		Significance	.043	.534	.005	.697	.045			.					
9	Configurability	Coefficient	.234	.146	.180	.020	.125				1.000				
		Significance	.000	.024	.005	.762	.053				.				
10	Standardization	Coefficient	.189	.144	.276	-.020	.261	.230	.200	.106	.070	1.000			
		Significance	.003	.026	.000	.764	.000	.000	.002	.100	.278	.			
11	Fewer functionalities	Coefficient	.080	.165	.135	.186	.141	-.060	-.008	-.068	.031	.143	1.000		
		Significance	.222	.012	.040	.004	.032	.354	.901	.297	.634	.028	.		
12	ln(Age)	Coefficient	-.056	-.076	.047	.063	.025	-.064	-.045	-.059	-.107	.010	.050	1.000	
		Significance	.387	.242	.475	.334	.697	.309	.469	.343	.085	.876	.444	.	
13	ln(Size)	Coefficient	-.127	-.033	.005	.043	.017	.154	.169	.040	.049	.114	-.102	.159	1.000
		Significance	.056	.621	.941	.519	.802	.016	.008	.539	.443	.086	.124	.009	.

model "DV=Scope" (outcome with many extreme values), Probit for the model "DV=Influence" (the underlying latent trait of the ordinal outcome is normally distributed) and Logit for the models "DV=Formula" and "DV=Discrimination" (evenly distributed categories). Finally, to test the proportional odds assumption the authors ran tests of parallel lines in SPSS. With all the models, the Chi-Square statistics were insignificant, indicating that the assumption was not violated.

4 Results

The variables and their non-parametric correlations are visible in Table 1. The results show that some variables capturing the changes in software firms' pricing models are positively correlated with the adoption of cloud technologies. Specifically, the change towards having simpler pricing model (Scope) is associated with multi-tenancy, online delivery, configurability; change towards less negotiation (Influence) is positively correlated with multi-tenancy and configurability; change towards usage-based pricing (Formula) is associated with virtualization, multi-tenancy, online delivery and configurability; and change towards more unified pricing across the customers (Discrimination) is associated with virtualization, multi-tenancy and online delivery. However, change towards shorter subscription periods (Temporal rights) is not correlated with the use of the technologies; thus, we exclude the ordinal regression model explaining this change by introducing cloud technologies from this study.

Change in value proposition toward more standardized product/service or towards fewer functionalities is associated with change in different pricing model elements. Table 1 also shows correlations between dependent variables.

Table 2. Ordinal regression models with parameter estimates

	DV=Scope			DV=Influence			DV=Formula			DV=Discrimination		
	Estimate	StdErr	Sig.	Estimate	StdErr	Sig.	Estimate	StdErr	Sig.	Estimate	StdErr	Sig.
DV ordinal level =1	-40.651	31.242	.193	-1.477	.527	.005	-.339	.919	.712	-.958	.990	.333
DV ordinal level =2	-.996	.945	.292	.073	.510	.887	.924	.895	.302	1.154	.913	.206
DV ordinal level =3	.749	.947	.429	1.060	.513	.039	2.684	.910	.003	2.760	.926	.003
DV ordinal level =4	4.911	1.306	.000	2.580	.544	.000	5.419	.965	.000	6.186	1.015	.000
virtualization	-.132	.285	.644	-.029	.164	.861	.218	.293	.456	.133	.294	.650
multi-tenancy	.943	.318	.003	.368	.171	.031	.557	.308	.071	.294	.309	.342
online delivery	.069	.304	.821	-.034	.182	.850	.425	.320	.184	.196	.323	.544
configurability	1.043	.311	.001	.155	.166	.351	.351	.297	.236	.226	.301	.453
standardization	.329	.140	.019	.101	.079	.199	.518	.140	.000	.394	.141	.005
fewer functionalities	.041	.143	.776	.252	.083	.002	.214	.145	.139	.264	.151	.080
ln(Age)	-.065	.186	.725	-.199	.104	.056	.180	.183	.325	.118	.187	.527
ln(Size)	-.081	.057	.153	-.009	.028	.753	-.037	.050	.457	.027	.051	.588
Pseudo R^2 (Nagelkerke)	.160			.115			.172			.098		
Pseudo R^2 (controls only)	.003			.016			.004			.004		
Model fitting information	536.509	35.630	.000	548.369	24.924	.002	543.388	38.424	.000	509.455	20.534	.008

Results from the ordinal regressions of the four models are shown in Table 2, which reports the regression parameter estimates for the levels of dependent variables ("DV"), for the independent variables and controls. The table also reports two pseudo r-squares of Nagelkerke – for the full model and for controls only – which assess the

overall goodness of fit of the ordinal regression models. While the values give some indication of the strength of the associations between the dependent and the predictor variables, the authors note that these r-squares should not be interpreted similarly to the OLS regressions. However, comparing the r-squares between a model including only controls and the full model, the higher r-square on each full model indicates better prediction on the outcome. Lastly, the tables include model fitting information for the final models; −2 log-likelihood, Chi-square and significance. The values are statistically acceptable for all models. This means that the models yield predictions more fitting than the marginal probabilities for the dependent variable categories.

Focusing on the ordinal regression parameter estimates for this study, the adoption of multi-tenancy is significant in predicting the change towards having simpler pricing model (in model "DV=Scope", Est.=943, Sig.=.003), towards less negotiation ("DV=Influence", Est. =.368, Sig. =.031) and to some extent notable in predicting the change towards usage-based pricing ("DV=Formula, Est. =.557, Sig. =.071). Besides, software firms with highly configurable applications are more likely to change their pricing model towards having simpler pricing model ("DV=Scope, Est. =1.043, Sig. =.001). The change towards simpler pricing model is also predicted by the standardization of the products and services ("DV=Scope", Est. =.329 Sig. =.019). Besides, change in value proposition towards more standardized product/service is a better predictor of changes towards usage-based pricing ("DV=Formula", Est. = .518, Sig.=.000) and toward more unified pricing across the customers (DV="Discrimination", Est.= .394, Sig.= .005) than the cloud technologies. Furthermore, change towards fewer new functionalities is the best predictor for change towards less negotiation (DV="Influence", Est.=.252, Sig.=.002).

5 Discussion

The current study supports most of our hypotheses deriving from the literature regarding the pricing model changes due to adoption of cloud computing technologies. The use of virtualization, multi-tenancy, online delivery and configurability are associated with the increased use of usage-based pricing. Besides, the use of multi-tenancy and configurability is associated with less negotiation with the customers. This can be explained by the fact that multi-tenancy constraints the customers' options for customization [51] that results in the customers' lower influence on both the product/service and its pricing. In addition to the above mentioned associations, the use of multi-tenancy, online delivery and configurability is significantly correlated with change towards simpler pricing with less pricing components. Also, the use of hardware virtualization, multi-tenancy and online delivery correlates with change towards more unified pricing across customers.

Based on the results, multi-tenancy is the most influential factor among cloud computing technologies that affects 4 out of 5 pricing model dimensions. Since multi-tenancy is the indicator of a cloud-mature, standardized application, it is not surprising that the use of it implies fundamental changes in the pricing as well. Prior research accentuates the role of multi-tenancy in the success of SaaS vendors [33].

However, based on this finding we claim that besides implementing multi-tenancy, changes most likely occur also in business model elements, such as the revenue logic, and these changes contribute *together* to the success. On the other hand, keeping our research method in mind, we cannot rule out the possibility that online delivery, configurability and virtualization might be introduced earlier than 3 years, leaving some dimensions of pricing models untouched during these last years.

It has to be noted that based on the empirical findings, the use of cloud computing technologies does not imply change towards shorter subscription contracts. Even though the use of these technologies enables shorter subscription contracts with the customers, the results show that the aim of software companies is to develop longer customer relationships. A possible explanation for this could be the possibly heavy competition in the market and the firms' high initial investments whose return need to be secured.

In the current study, besides technological characteristics and changes in pricing model elements, our model incorporated also changes towards more standardized products/services and fewer functionalities. The results show that change in value proposition explains most of the changes in different pricing model elements. This underscores the interrelation of different business model elements suggested by the literature (e.g. [10], [47]); namely, decisions to individual business model elements may affect several aspects of the firm.

Firms that standardize their products and services change also their pricing model; thus, revenue logic is highly important in a firm's strategy that needs attention from the managers. Besides standardizing the software, unifying the pricing across customers and using more volume dependent pricing components is justified. Standardized, less customer-specific software can be sold for the same price for different customers since the minimal customization work offsets the differences in the development costs. Standard software may generate more revenues with employing usage-based pricing in case there are big differences in the users' demand. With incorporating usage-dependent pricing components into the revenue logic, the infrastructure costs are passed directly from the provider to the customers. This way the company is able to catch also the long-tail of the market.

The analysis shows also that companies that implement fewer new functionalities give less negotiation power to their customers. Concentrating on the core functionalities leaves no or minimal room for user-specific customization work, thus, it makes negotiation unnecessary. Hence, SaaS firms offering standard software with a limited set of core functionalities usually employ pricelists in their pricing to attract customers.

The strength of associations between variables in this study indicates that implementing technological and business models changes is complex. The software firm's managers' cognitive processes may play an important role in adjusting different business model elements, in some cases even greater than the technological opportunities. We consider also the possibility that the software firm had already executed the changes before, thus, there had not been changes in the last three-year period.

During the study, we paid special attention to the common possible bias in survey research, such as measurement errors, problems related to sampling, coverage, and non-response [44]. To reduce the risk of measurement error we attained guidance on the survey questions from both researchers and practitioners in the field. Whenever available, we applied scales that have been tested in previous studies. One of the concerns with the measurements is the use of single-item measures, which are argued to insufficiently capture the conceptual domain. However, this claim has been challenged by DeVellis [52] by arguing that each item of a scale is precisely as good measure as any other of the scale items and that the items' relationship and errors to the variable are presumed identical. Understanding of this perplexity guided the authors not to make claims about the changes in pricing model dimensions (e.g. scope of the pricing model), but rather about the parameters (e.g. the number of pricing model components).

The software industry survey practically covers and contacts all the Finnish software companies; therefore we consider coverage and sampling errors irrelevant. The overall sampling rate for the software industry survey nonetheless is roughly 10 percent, which suggests a potential risk of non-response bias. However, the effective sample contained software firms of all types, ages and sizes, and the concern is principally if there are theoretically relevant differences between respondents and non-respondents. In this case, the effective sample contained sufficient variety in dependent variables to support the analysis of the hypotheses.

6 Conclusions

Using cloud computing technologies in software applications implies changes also to the business aspects of software firms; among which pricing is extremely important in achieving success in the competitive SaaS market. The current study fills a research gap in the current literature by focusing on the impact of deploying cloud computing technologies on different pricing model elements. In this paper the results of the research are presented related to the impact of hardware virtualization, multi-tenancy, online delivery and configurability on different dimensions of pricing models, such as the scope of it, the influence of the customers on pricing, the use of usage-based pricing, the temporal rights and price discrimination across customers.

After analyzing an effective sample of 324 software firms, we conclude that the use of cloud computing technologies implies changes in different dimensions of the pricing models. The results show that multi-tenancy is the most influential factor, affecting 4 out of 5 dimensions, while hardware virtualization, online delivery and configurability are associated with changes in some of the aspects of the pricing model. Software firms that use cloud computing technologies in their products and services seem to make their pricing model simpler, use usage-based pricing, reduce the customers' influence and unify their pricing across customers. They do not, however, shorten the length of the contracts with their customers. The current study also revealed that changes in pricing models happens together with changes in the value proposition; this underlines the interrelation of different business model elements suggested also by the literature (e.g. [10], [47]).

This study is the first to examine the changes in pricing models of SaaS firms empirically and therefore the authors suggest these findings to serve as a starting point for future studies. The practical implication of this study is an increased understanding about how the SaaS vendors are changing their business models and consequently how the market of software products and services is evolving as a result of recent technological advances. As the market is transforming to embrace the promises of cloud computing technologies, studies on business models offer predictions about what are the viable configurations of business models and how deployment of technologies changes the configurations. Since the survey is limited to Finland, the study does not necessarily provide a representative illustration on SaaS firms in a global context; therefore similar studies in other countries are welcome to complement the results.

References

1. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud computing—The business perspective. Decision Support Systems 51(1), 176–189 (2011)
2. Chong, F., Carraro, G.: Architecture strategies for catching the long tail. MSDN Library, Microsoft Corporation, pp. 9–10 (2006)
3. Weinhardt, C., Anandasivam, D.-I.-W.A., Blau, B., Borissov, D.-I.N., Meinl, D.-M.T., Michalk, D.-I.-W.W., Stößer, J.: Cloud computing–a classification, business models, and research directions. Business & Information Systems Engineering 1(5), 391–399 (2009)
4. Benlian, A., Hess, T.: Opportunities and risks of software-as-a-service: Findings from a survey of IT executives. Decision Support Systems 52(1), 232–246 (2011)
5. Choudhary, V.: Comparison of software quality under perpetual licensing and software as a service. Journal of Management Information Systems 24(2), 141–165 (2007)
6. Stuckenberg, S., Fielt, E., Loser, T.: The impact of software-as-a-service on business models of leading software vendors: experiences from three exploratory case studies. In: Proceedings of the 15th Pacific Asia Conference on Information Systems, PACIS 2011 (2011)
7. Tyrväinen, P., Selin, J.: How to sell saaS: A model for main factors of marketing and selling software-as-a-service. In: Regnell, B., van de Weerd, I., De Troyer, O. (eds.) ICSOB 2011. LNBIP, vol. 80, pp. 2–16. Springer, Heidelberg (2011)
8. Gourville, J., Soman, D.: Pricing and the Psychology of Consumption. Harvard Business Review 80(9), 90–96 (2002)
9. Piercy, N.F., Cravens, D.W., Lane, N.: Thinking strategically about pricing decisions. Journal of Business Strategy 31(5), 38–48 (2010)
10. Teece, D.J.: Business models, business strategy and innovation. Long Range Planning 43(2), 172–194 (2010)
11. Anandasivam, A., Premm, M.: Bid price control and dynamic pricing in clouds. In: ECIS 2009 Proceedings (2009)
12. Schramm, T., Wright, J., Seng, D., Jones, D.: Six questions every supply chain executive should ask about cloud computing. Accenture Institute for High Performance (2010)
13. Cusumano, M.A.: The changing labyrinth of software pricing. Communications of the ACM 50(7), 19–22 (2007)
14. Magretta, J.: Why business models matter. Harvard Business Review 80(5), 86–92 (2002)

15. Osterwalder, A., Pigneur, Y., Tucci, C.L.: Clarifying business models: Origins, present, and future of the concept. Communications of the Association for Information Systems 16(1), 1–25 (2005)
16. Baden-Fuller, C., Morgan, M.S.: Business models as models. Long Range Planning 43(2), 156–171 (2010)
17. Casadesus-Masanell, R., Ricart, J.E.: From strategy to business models and onto tactics. Long Range Planning 43(2), 195–215 (2010)
18. Demil, B., Lecocq, X.: Business model evolution: in search of dynamic consistency. Long Range Planning 43(2), 227–246 (2010)
19. Chesbrough, H., Rosenbloom, R.S.: The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies. Industrial and Corporate Change 11(3), 529–555 (2002)
20. Cavalcante, S., Kesting, P., Ulhøi, J.: Business model dynamics and innovation (Re) establishing the missing linkages. Management Decision 49(8), 1327–1342 (2011)
21. Bharadwaj, A., El Sawy, O.A., Pavlou, P.A., Venkatraman, N.: Digital Business Strategy: Toward a Next Generation of Insights. MIS Quarterly 37(2), 471–482 (2013)
22. Kamoun, F.: Rethinking the business model with RFID. Communications of the Association for Information Systems 22(1), 35 (2008)
23. Timmers, P.: Business models for electronic markets. Electronic Markets 8(2), 3–8 (1998)
24. Wirtz, B.W., Schilke, O., Ullrich, S.: Strategic development of business models: implications of the Web 2.0 for creating value on the internet. Long Range Planning 43(2), 272–290 (2010)
25. Ojala, A., Tyrväinen, P.: Developing Cloud Business Models: A Case Study on Cloud Gaming. IEEE Software 28(4), 42–47 (2011)
26. Novelli, F.: A mixed-methods research approach to investigate the transition from on-premise to on-demand software delivery. In: Vanmechelen, K., Altmann, J., Rana, O.F. (eds.) GECON 2012. LNCS, vol. 7714, pp. 212–222. Springer, Heidelberg (2012)
27. Campbell-Kelly, M.: Historical reflections: The rise, fall, and resurrection of software as a service. Communications of the ACM 52(5), 28–30 (2009)
28. Cusumano, M.A.: The changing software business: Moving from products to services. Computer 41(1), 20–27 (2008)
29. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Gaithersburg, MD: National Institute of Standards and Technology (2011)
30. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, GCE 2008, pp. 1–10 (2008)
31. Bezemer, C.-P., Zaidman, A.: Multi-tenant SaaS applications: maintenance dream or nightmare? In: Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), pp. 88–92 (2010)
32. Bezemer, C.-P., Zaidman, A., Platzbeecker, B., Hurkmans, T., Hart, A.: Enabling multi-tenancy: An industrial experience report. In: 2010 IEEE International Conference on Software Maintenance (ICSM), pp. 1–8 (2010)
33. Guo, C.J., Sun, W., Huang, Y., Wang, Z.H., Gao, B.: A framework for native multi-tenancy application development and management. In: Proceedings of CEC/EEE 2007, pp. 551–558 (2007)
34. Ju, J., Wang, Y., Fu, J., Wu, J., Lin, Z.: Research on key technology in SaaS. In: International Conference on Intelligent Computing and Cognitive Informatics (ICICCI), pp. 384–387 (2010)

35. D'souza, A., Kabbedijk, J., Seo, D., Jansen, S., Brinkkemper, S.: Software-As-A-Service: Implications For Business And Technology in Product Software Companies. In: PACIS 2012 Proceedings (2012)
36. Ojala, A.: Software renting in the era of cloud computing. In: IEEE 5th International Conference on Cloud Computing, pp. 662–669 (2012)
37. Ojala, A.: Revenue models in SaaS. IT Professional 15(3), 54–59 (2012)
38. Lehmann, S., Buxmann, P.: Pricing Strategies of Software Vendors. Business & Information Systems Engineering 1(6), 452–462 (2009)
39. Iveroth, E., Westelius, A., Petri, C.-J., Olve, N.-G., Cöster, M., Nilsson, F.: How to differentiate by price: Proposal for a five-dimensional model. European Management Journal 31(2), 109–123 (2013)
40. Laatikainen, G., Ojala, A., Mazhelis, O.: Cloud Services Pricing Models. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 117–129. Springer, Heidelberg (2013)
41. Turner, M., Budgen, D., Brereton, P.: Turning software into a service. Computer 36(10), 38–44 (2003)
42. Laplante, P.A., Zhang, J., Voas, J.: What's in a name? Distinguishing between SaaS and SOA. IT Professional 10(3), 46–50 (2008)
43. Lehmann, S., Draisbach, T., Buxmann, P., Dörsam, P.: Pricing of Software as a Service – An Empirical Study in View of the Economics of Information Theory. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 1–14. Springer, Heidelberg (2012)
44. Dillman, D.A.: Mail and internet surveys: The tailored design method. Wiley, New York (2000)
45. Zott, C., Amit, R., Massa, L.: The Business Model: Recent Developments and Future Research. Journal of Management 38(1), 375–414 (2011)
46. Miller, D.: Configurations of strategy and structure: Towards a synthesis. Strategic Management Journal 7(3), 233–249 (1986)
47. Hacklin, F., Wallnöfer, M.: The business model in the practice of strategic decision making: insights from a case study. Management Decision 50(2), 166–188 (2012)
48. McCullagh, P.: Regression models for ordinal data. Journal of the royal statistical society. Series B (Methodological), 109–142 (1980)
49. Podsakoff, P.M., MacKenzie, S.B., Lee, J.-Y., Podsakoff, N.P.: Common method biases in behavioral research: a critical review of the literature and recommended remedies. Journal of Applied Psychology 88(5), 879 (2003)
50. Norusis, M.: SPSS 16.0 guide to data analysis. Prentice Hall Press (2008)
51. Xin, M., Levina, N.: Software-as-a-service model: Elaborating client-side adoption factors. In: Proceedings of the 29th International Conference on Information Systems (2008)
52. DeVellis, R.F.: Scale development: Theory and applications. Sage (2011)

What Influences Platform Provider's Degree of Openness? – Measuring and Analyzing the Degree of Platform Openness

Anisa Stefi, Matthias Berger, and Thomas Hess

Ludwig-Maximilians-Universität München, Institute for Information Systems and New Media, Ludwigstrasse 28, 80539 Munich, Germany
{stefi,matthias.berger,thess}@bwl.lmu.de

Abstract. Software platform providers are increasingly opening up their platforms to take advantage of external resources and innovations. This research, based on existing theories, empirically analyzes the drivers of platform providers' decision to open up their platforms. Based on resource-based view and control mechanism, we developed a set of hypotheses which explain the degree of platform openness adopted by the platform providers. We performed qualitative interviews and quantitatively validated the results. Thus, we developed an online survey targeting software platform providers. The results show that absorptive capabilities of platform providers play an important role in the degree of platform openness. We also found that the implementation of informal control modes influence the decision to use a higher degree of platform openness.

Keywords: Software Platform, Degree of Openness, Resource-based View, Agency Theory.

1 Introduction

In recent years, we have witnessed the success of software platforms opening up and taking advantage of external resources, expanding their provided functionalities through apps or plug-ins. The main technology, the software platform, is defined as "the extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate" [1 p.1]. One of the best known examples - mobile apps - generate revenues that were estimated to be around $26 billion in 2013 [2]. An example in the area of enterprise software is Salesforce, which first allowed companies to build applications utilizing features offered in its Customer Relationship Management product through AppExchange, then further extended this by providing SalesForce.com, which offers a service platform on which other companies can build their own applications based on Salesforce software [3]. Opening up an existing platform, can provide benefits due to the additional functionalities offered by external modules, which address a great number of user needs. However, it can also increase

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 258–272, 2014.

competition and lower switching costs making it difficult for the platform provider to generate revenues [4]. Therefore, the decision to open up an internal software platform and identifying the right degree of openness is of strategic importance for a software platform provider, since it affects the business model of the whole company. The concept of platform openness has been the focus of several studies in the information system (IS) research [4, 5]. A purely open platform is defined as one where there is no restriction to its participation in its "use, development and commercialization" [6 p.1851]. The opposite of open is a closed system defined as "wholly owned, proprietary, vertically integrated, and controlled by a single party" [6 p.1851]. Some studies consider software openness as a dichotomous response, being either closed or open [7], but research has recognized that there are various degrees of being open, thus acknowledging the degree of openness as more adequate measurement [6, 8].

In this study, we view the degree of openness as a continuum ranging between open and closed software. Whereas research has focused on how the perceived degree of openness affects external developers [9] or innovation [6], little is known about the drivers that influence software platform providers to adopt a certain degree of openness [1]. This is also shown by the lack of measurements to analyze the degree of openness of software platforms. Therefore, we argue that the strategic decision to provide a certain degree of openness to attract external developers needs to be carefully analyzed. Grounded on existing theories, this research analyzes the factors that affect the degree of openness. We aim at contributing to this research gap by addressing the following question:

What factors influence the degree of software platform openness from a platform provider's perspective?

In order to address this question, we adopt concepts considering both dynamic innovation capabilities of the providers as well as their concerns regarding the loss of control. We further provide a set of hypotheses which we test empirically.

The remainder of the paper is structured as follows. In the next section we will present the theoretical background. In section three, the hypotheses and the research model will be derived. The design of the empirical study follows in the fourth section. The fifth section will present the data analyses, which are further discussed in the sixth section. We conclude the paper by addressing the limitations and future work.

2 Theoretical Background

2.1 Platform Openness

Internal software platforms are one of the most successful forms of intra-organizational software reuse [10]. The relationship between software platform providers and the external developers or companies contributing with external components, plug-ins or extensions, creates an ecosystem around this platform [11]. The author in [12] argues that companies should carefully consider opening up their platform in order to create an ecosystem that could increase the value of the core offering, attract new users, share the innovation cost and integrate the functionalities

created by other partners. They further state that platform leaders need to determine the boundaries of the platform, especially the degree to which the product is made in-house and properly organize internal teams to prevent conflicts with external developers [12]. Platform openness is also analyzed by [13], who examined three practical case studies with mixed, proprietary and open strategies. The author argues that the use of a completely proprietary approach is only suitable and feasible for very few market leaders, whereas most companies tend to open up their platforms for common innovation despite losing differentiation opportunities [13]. According to the authors in [6], granting access to external developers leads to a trade-off situation in which control over the platform is lost, but the acceleration of the technology's diffusion process increases [6]. In their strategic analysis of platform openness, [4] point out that the overall strategy reflects which actors are restricted and which platform parts are opened or not [4]. By focusing on the perspective of complementary developers, [9] quantitatively found out that developers' perceived platform openness influences the satisfaction of the external developer with the platform. From a provider perspective, [14] qualitatively analyzes factors that influence the platform providers to adopt an open strategy. Although the concept of platforms openness has been widely discussed in the literature, more empirical research is needed to better understand the degree of openness (DOO) from the platform provider perspective, where a measurement for the DOO is still missing.

2.2 Resource-Based View and Agency Theory

Due to the study's explorative character, we focus on two theoretical approaches in order to explain the degree of platform openness, resource-based view (RBV) and agency theory because they concern the use of firm resources and the collaboration with external parties.

RBV theory define firm resources as all assets, such as organizational processes, knowledge or information that serve as enablers of effective and efficient strategies [15]. The combination of resources in form of input flows creates firm-specific capabilities. Those consist of high-level routines which provide a company's management with different decision options for deploying the capabilities [16]. The focus on a firm's own capabilities can explain the transfer of partial software development actions to external developers. Hence, the lack of required competences can be compensated through the acquisition of external resources [17]. This argumentation is also used in open innovation theory which deals with the company's strategy to connect in-house resources and external ideas for market commercialization through flexible boundaries [18]. Both absorptive capabilities and innovative capacities are required to successfully acquire, transform and integrate external knowledge into internal resources [19, 20]. For example, open source as external knowledge source, is adequate if the commonly shared technological innovations can be integrated to produce complex software systems [21]. Platform providers implement the open innovation concept by opening the internal software product line to entwine in-house research and development with the ideas of outside developers [22]. Functioning as an enabler for cooperation, a software platform makes this joint innovation creation possible. The opening decision reflects the strategic

importance of external resources for creating value for the end-user, which would not be possible acting independently from other participants [23]. Thus, the decision to open the platform should empower the provider to utilize the cooperation to generate new ideas and to incorporate them in the platform [10].

The new relationships that emerge by opening an internal platform also entail certain risks. Agency theory explains relationships, in which tasks are assigned from principals to agents (e.g. outsourcing projects) [24, 25]. Due to information asymmetries and individual profit maximization, in such contractual relations, main challenges arise if both parties have divergent goals or if the principal lacks the possibility to monitor the agent's actions and efforts [25]. As a consequence, it is essential for the principal to choose adequate control mechanisms to reduce information asymmetry, as it allows coordinating the behavior and verifying agents' actions [26, 27]. In general, two basic control strategies are at the principal's disposal. The first type is constituted by formal control modes: outcome control assesses to which extent determined goals are met and process control includes the specification of actions to guide the outsourced projects [28, 29]. The second control strategy refers to the informal control modes, clan control and self-control, both relying on the controllee's engagement to behave in the controller's interest because of either common values in the first case or internal norms and motivation in the latter one [29, 30]. These control mechanisms in principal-agent like relationships is relevant when software development activities are overtaken by parties other than the focal developer coordinating the development process. For example, in open source projects, a comprehensive and careful mix of formal, clan and self-control seems to be crucial when coordinating external developers [31]. In the context of software platforms although there are no contract relationship, the provider can adopt similar control modes so that complementary developers can add value to the platform based on joint objectives [32].

3 Research Model and Hypotheses

The decision to open up the platform is a strategic decision that requires the firm to identify which resources could be better addressed by external providers such as the open source community. Platform providers can thus concentrate on their core competencies and leverage specialized software parts of complementary actors [33]. Additionally, platform providers will want to integrate the innovative ideas coming from external parties. Therefore the dynamic capabilities of the platform provider such as absorptive and innovative capacity would influence the DOO.

As previously mentioned the decision to open up a platform also involves some risks and brings up new relationships which require the use of different control modes by platform providers. In analogy to the principal-agent relationship, the platform provider monitors the platform-related development actions also acting as a controller of the complement developers (controllees) [34]. Therefore, availability of formal control modes (process control), and informal control (clan control and self-control) would influence the DOO adopted from the platform provider [1]. Based on the above, we formulate the following hypotheses.

Absorptive Capacity
One of the reasons identified in the literature that motivates firms to open an internal platform is to exploit external knowledge [10, 14]. In order to be able to take advantage of the external resources, the firm needs to have the capability to assimilate, absorb and use the externally created resources [20]. Firm's ability to recognize and assimilate external information and to apply it to commercial ends is referred to as absorptive capacity [35]. Therefore, a platform provider which has a high absorptive capacity will be more willing to open up its platform and integrate external functionalities in the core platform. Hence, we posit the following:

H $_1$: *The higher the level of platform providers' absorptive capabilities is, the higher the degree of openness of the platform will be.*

Innovative Capability
A benefit of opening up an existing platform is the ability to profit from the provision of complementary assets [13]. This possibility enables providers to explore new innovative ideas that could not be implemented otherwise. Innovative capabilities enable companies to create and commercialize innovative products or processes [36]. Thus, they include two different dimensions: either they concern the ability to develop new ideas or the capacity to transform ideas into marketable goods [19, 37]. Hence, radical innovative capabilities are needed to come up with completely new products whereas incremental innovative capabilities help to strengthen the market potential of already existing ideas [38]. The stronger the innovative capabilities of the firm through internal development, the less likely the firm is to open up its platform to gain access to external innovative ideas as it can develop such ideas in-house. Based on the above, we formulate the following hypothesis.

H$_2$: *The higher the level of platform providers' innovative capabilities is, the lower the degree of openness of the platform will be.*

Modularity
According to the authors in [39], "a complex system is said to exhibit modularity in design if its parts can be designed independently but will work together to support the whole" [39 p.1117]. Similarly, Tiwana [34] refers to modularity as "the degree of intentional decoupling among constituent subsystems" [34 p.771]. In the context of open source, modularity was one of the main product characteristics that contributed to foster the participation of distributed developers [40]. Therefore, as it exhibits an "embedded coordination mechanism" [34 p.6], modularity reduces the need for managerial supervision and therefore presents an effective substitute for process control [34, 41]. Platform providers rely on such a process control mode to coordinate and accomplish the transfer of standards, specifications and guidelines to external component developers [32]. Thus, a higher level of modularity offers a higher level of process control to software providers, which allows them to better control the development of their platform. Considering the above argumentation, we can state the following hypothesis.

H$_3$: *The higher the level of platform modularity is, the higher the degree of openness of the platform will be.*

Clan Control
Clan control is one of the informal control modes that is normally implemented by narrowing the gap between the controller's and controllee's preferences based on shared values or goals [30]. In software development, this informal mode uses common norms and interests and mutual monitoring to align the effort and action put into the development of a project [31]. Consequently, clan control empowers users to exchange tacit knowledge with other participants because this implicit know-how often is internalized in the shared values and norms [32]. Therefore, with an increasing extent of clan control, platform providers are able to adopt a higher DOO since it offers further options to control the development of the software platform.

H_4: *The higher the level of clan control encouraged by the platform provider is, the higher the degree of openness of the platform will be.*

Self-control
The second informal control mode, the self-control, is implemented when the controllee is motivated to act in the controller's interest, because of internal motives, by determining own objectives in line with the overall platform's goals [30]. Especially in open software development, self-motivation is important in order to come up with innovative ideas [31]. Self-control reduces the need of central control through a reciprocal adaption and adjustment between platform shaper and external actors [42]. The behavior of all participants is based on the commitment to follow the common business vision [cf. 43]. Therefore, self-control positively influences the DOO adopted by the provider since it decreases the necessity of strict monitoring by aligning all the participant's goals. Thus, the following can be posited.

H_5: *The higher the level of self-control modes encouraged by the software provider is, the higher the degree of openness of the platform will be.*

Degree of Openness
The DOO describes how open a platform is for the integration of externally developed resources. As recommended in [8], it consists of five dimensions: Availability, Accessibility, Transparency, Reciprocity and Licensing. Availability describes the feasibility of opening up the platform. Accessibility reflects the extent to which the technology of the platform enables complementary players to contribute with their own components. Transparency describes how functions and processes of the platform are transparent for the users. Reciprocity describes the reciprocal interactions between platform and complement developers with the aim of a bi-directional information exchange. Licensing refers to the use or redistribution of software rights.

Research Model
Our hypotheses are summarized in the following research model.

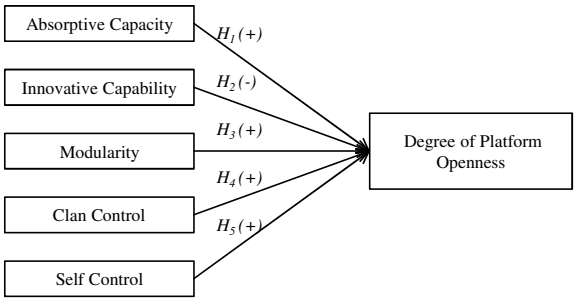

Fig. 1. Research Model

4 Research Design

We conducted expert interviews to ensure the comprehensiveness of the research model and further validated the hypotheses quantitatively through an online survey.

4.1 Qualitative Pre-study

To ensure content and substantive validity we conducted six expert interviews. The interview partners were selected by first searching a firm database for firms that have a platform and by identifying a contact person through professional social networks (e.g. LinkedIn). The participants had at least 5 years of professional experience in software development and a current position of at least team leader. They were in some way involved with software platforms and their companies covered different types of more open/closed software systems and therefore they could provide insights on different open strategies. Interview questions considered the relevance of research question, the feasibility of the model as well as the measurement of the related constructs [44]. The questions addressed the interviewees' opinions regarding the perception of openness of their product lines, its critical drivers as well as what dimensions are necessary to measure the DOO. In the end, the research model and the scale items of the operationalized DOO measure were shown to the experts for evaluation, checking for completeness and reliability.

We performed a content analysis of the interviews which is conducted by first assessing how statements match the deductively derived categories in the model. Secondly, the inductively recognized categories are classified on a higher abstraction level [cf. 45]. From the interviews, three central patterns appear across all results. Firstly, all interview partners regard the decision to open up a software platform as highly strategic, as it facilitates the access to new market segments and provides flexible reactions to consumers' needs. Secondly, innovation seeking is pointed out as a crucial motivation factor for integrating external knowledge. Regarding the control modes, their meaning as critical drivers is stressed at some extent. Thus, several interviewees see modularity as an effective instrument to implement process control and also describe some forms of clan control between platform providers and complement developers. Finally, the key attributes of DOO were found relevant for

measuring the degree of platform openness and were slightly adapted based on the interviews. An expert from a free open source project stated that all items would suggest openness for his project, which is a good indication for the content validity.

4.2 Model Operationalization

The dependent variable, the DOO is measured on the basis of its five attribute dimensions (see Section 3). To provide a percentage degree of the DOO measure, every item of this construct is categorized as an "open" aspect, coded with 1, or as a "closed" aspect, coded with 0, following the recommendation in [8]. These openness points are then accumulated and divided by the total number of all items. Unless otherwise specified, all indicator statements of DOO are evaluated on a seven-point Likert scale with anchors ranging from "completely correct" to "not a bit correct". The participants could also select "no statement possible". The items of the five dimensions were adapted to the platform provider context from existing studies. The missing dimensions were developed based on existing literature and the qualitative interviews (see Table 1). *Availability* provides items that describe the availability of resources (e.g. code, data or APIs) and if technical barriers prevent the delivery of these resources [6, 46]. In analogy to [9], *Accessibility* indicators regard the easiness of learning technological standards, technical interoperability and developers' freedom to select standards or interfaces [9]. Based on the qualitative interviews, accessibility is viewed as granting access to outsiders and considers the relevance of programming languages and of platform integrated data processing. *Transparency* involves the disclosure of technical platform features and related organizational and managerial processes. In analogy to [9], respective indicators embed the completeness of documentation, the implementation of communication channels and supportive knowledge sharing. *Reciprocity* facilitates knowledge contribution and sharing, which is motivated by mutual benefits from a bi-directional information and experience exchange [8, 46]. Finally, *Licensing* explains different variations in software openness due to the number of outsiders who are granted access under certain conditions [6]. This construct is measured on "Yes" or "No" scale and has a major importance with six possible openness points. Thereby, it should be noted that licensing was mentioned by all experts' interviews. The items of the independent constructs are adapted from existing studies (see Table 2).

4.3 Study Design and Sample

The data was collected through an online survey. The pre-testing phase led to changes in wording and the order of questions. Additionally, respondents were required to state the actual *openness status*, i.e. if their software platform is implemented as OSS or provides an API for integration. In the end of the survey demographics and control variables such as expertise and the management position in the firm were asked. Respondents were prompted to answer the questions based one specific software project they were working on or that is representative for their organization [47]. An intensive search through firm databases generated a set of 526 software companies

that stated having an already implemented platform. Therefore, 252 personalized invitations were sent via e-mail to software managers or heads of development. The remaining 274 companies were contacted with an invitation to their general mail addresses, containing a request to pass the mail on to the head of development.

Table 1. Operationalization of the Degree of Openness

Scales	Source	Adapted from
Availability	[6, 46]	Our company makes resources available to 3-rd party developers. No technical barriers hinder the delivery of resources/components/tools to 3rd-party developers
Accessibility	[9]	Our platform was designed in such a way that enables external developers to easily learn its technical standards. Technically, we configured the platform to allow interoperability with other systems or platforms. The external developers' freedom to choose standards, interfaces, programming languages is not restricted by the functional range of our platform. The programming language used is highly specific for our platform. The data generated with our platform is compatible with other platforms.
Transparency	[9], [48]	The provided technical documentation of the platform is comprehensible, useful, and complete. The platform promotes features to encourage communication and exchange with our company. We share our experience and knowledge with other organizations from the platform network. Procedures and processes to integrate 3rd-party components are highly formalized. Third-party developers are informed about decision-making processes regarding our platform.
Reciprocity	[49]	We expect developers to contribute value to our platform, so it's fair to help them with our knowledge. We should feel an obligation to contribute with our knowledge because we use the knowledge from the platform and external developers. If we receive useful knowledge from the 3-rd parties, we should provide useful knowledge to others in return Our organization wants to help every partner, if their external software components are useful.
License	[6], [50]	We follow a policy of not licensing 3rd-party developers. We permit only a restricted number of licenses and restrict these to development of products for market niches. We permit only a restricted number of licenses, yet, without restrictions to particular market segments. We allow all reliable 3rd-party developers to integrate components. Generally, the licensing of 3-rd parties produces technical restrictions. Generally, the licensing of 3-rd parties produces commercial restrictions.

Table 2. Operationalization of the Independent Constructs

Scales	Items	Adapted from	Scale
Absorptive Capacity (ACAP)	3	[51, 52]	7-Point Likert
Innovative Capability (ICAP)	6	[38]	7-Point Likert
Modularity (MOD)	4	[34]	7-Point Likert
Clan Control (CCON)	4	[53, 54]	7-Point Likert
Self-Control (SCON)	4	[47, 53]	7-Point Likert

A total of 47 completed responses were collected, yielding a response rate of 8.94%. Survey participants were between 31 and 62 years old and 75% of them had at least 11 years of experience in the software industry. Most of the interviewers had a college or university degree (75.6%) and 14.7% had a doctorate degree. The data included respondents from the top management (26.67%), the middle management (40%) and the lower management (22.22%). The number of employees in an organization varied, with organizations with less than 50 employees (50%), between 50-100 (4.44%) and more than 100 employees (35.56%). On average, an organization was working on 6.79 development projects. The relatively small number of respondents is acceptable, for the explorative purpose of this study, since according to [55], a number of respondents higher than twice the number of variables is sufficient for regression applicability.

5 Data Analysis

5.1 Instrument Validation

In analogy to [9], the DOO of a platform is interpreted as the sum of the single attributes. Before we can calculate the DOO scale, we first need to validate and verify the measurement of its dimensions: *Accessibility*, *Transparency* and *Reciprocity*, to assess if the indicators fit these latent constructs within the overall scale [55]. For *Availability* and *Licensing* this procedure is not necessary because they do not imply the application of latent construct, but can directly indicate openness in their respective items. Since values for the Kaiser-Myer-Olkin (KMO) criterion and the variable-specific of measures of sampling adequacy (MSA) criterion do not contradict, factor analysis is applicable for the three former dimensions. Factor loadings of the respective items exhibit values between 0.635 and 0.856 and communalities values are above the threshold of 0.50 [cf. 55]. According to [55], all measures exhibit good item reliability and items seem to accurately quantify specific DOO dimensions. The analysis of explained total variance and factor loadings leads to a slight re-arrangement of items within the dimensions. Yet, this has no influence on the DOO measure because statements about the openness points do not depend on the allocation of the items. Furthermore, the psychometric properties measures of the average variance extracted (AVE) and composite reliability verify the validity of individual indicators, convergent validity of extracted factors and discriminant validity, since values lie above 0.50 and 0.70 respectively [55].

To calculate the DOO, the indicators of all five attribute dimensions that designate an open facet are coded as 1. This includes the Likert scale answer categories 5 to 7

and in the case of licensing, the "yes" or "no" category that reflects an open state. The DOO is calculated as described in Section 4.2. The DOO ranges from 5.0% to 100.0% with a mean value of 53.0%. Median (0.55) and skewness (0.95) together with the results of Kolmogorov-Smirnov test (KS-Z = 0.496; p-value = 0.966) allow to assume a normal distribution and item-to-total correlations for the respective items, internal homogeneity and overall item reliability can be confirmed. Content validity is assessed as constructs have been applied in other studies (see Table 2). To evaluate the reliability of the independent variables' constructs, internal consistency measures are calculated. Cronbach's alphas are above the critical value of 0.70 for all five independent constructs and high enough item-to-total correlations for the respective items, internal homogeneity and overall item reliability can be confirmed. Thus, the variables are adequate for further analyses. We calculate a combined score of the independent constructs by calculating the average of their Likert scale values. This score represent their corresponding variable and can be used in a multiple regression model or correlation analysis.

5.2 Analysis of the Research Model

We examine the relationships between the factors described above and the *DOO* by performing a multiple regression analysis. The required regression assumptions of linearity, homoscedasticity and normal distribution are met since values for the tolerance statistic, the variance inflation factors as well as the range and plotting of residuals confirm the applicability of regression analysis [55]. The linear regression analysis shows that the five independent variables explain 53.0% (R^2-value) of the variance in the dependent variable. According to [55], the significance of the estimated model is acceptable (F-value = 9.252; p-value = 0.000). The results show that *Clan control* exhibits the strongest, highly significant influence on DOO (ß = 0,064, t-value= 3.131). *Absorptive capability* also affects the dependent variable positively with 0.034 as coefficient value, significant on the 5%-level (t-value= 2.191). Finally, a weakly significant, positive relation is observed for *Self-control* (ß = 0.036; t-value = 1.715). Due to their lack of significance, the other two variables, *Modularity* and *Innovative capability,* do not show a linear influence on the *DOO*. However, the coefficients have the algebraic signs suggested in the hypothesis. Thus, from the suggested research model, confirms H1, H4 and slightly support H5.

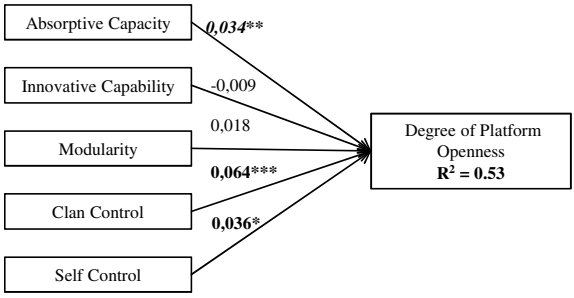

Fig. 2. Regression Analysis

6 Discussion

The study provides to two main contributions, the results of the regression model and appropriateness of the DOO measure used in this study. The results of the regression analysis further highlight that the platform provider's ability to absorb external knowledge has a positive and significant impact on the openness level of the software platform. Accordingly, the existence of strong absorptive capabilities could lead to a high DOO. This finding corresponds well with overall goals of software platform providers. If platform providers are unable or unwilling to develop specific functionalities, externally programmed software parts and components can be used to extend the utility and the functional range of the platform [10]. Furthermore, it is essential for the evolution of the software platform to continuously embed new functionalities into its core functions. Thus, platform providers should establish good connections to the other actors, which can support in innovation searching and seeking [22, 43]. Different than expected, no significant relationship is identified between innovative capabilities and the DOO. Although the suggested influence direction is met in the regression, the second hypothesis has to be rejected. A possible explanation could be that innovative capabilities are not only concerned with the ability to come up with new software parts or processes for the enhancements of the platform but also to bring ideas into marketable products [37]. This second aspect could positively contribute to the DOO as providers can benefit from external ideas.
Modularity as substitute for process control had no significant linear influence on DOO. A possible explanation is that for the mere integration of functionalities, the availability of interfaces is sufficient. In contrast, the utilization of informal control modes can have a major positive impact on the DOO. Thus, a higher extent of clan control or self-control causes a higher DOO for the platform. Thus, due to the different significance levels, hypothesis four can be strongly confirmed and hypothesis five weakly confirmed suggesting that both control modes are critical drivers. The realization of clan control and self-control entails that responsibilities for the orchestration of software development are shared between the platform and the complementary developers [42].

The validation of the DOO scale highlights its usefulness and applicability to quantify a certain level of software platform openness. The scale also reflected the desired continuum [5, 8], a distribution over almost the complete range of possible values between a purely closed and purely open state. Thus, existing open systems still have some constraints in their use. The surveyed data shows that reciprocal licenses which pose special obligations on combinations of software components to allow reuse and sharing [50], is often observed in OSS projects. Documentation is another aspect, that as mentioned by one expert interview, badly documented OSS cannot be denoted as completely open. For platform providers, these findings mean that they have a variety of opportunities and decisions regarding their platform governance. They are able to choose a certain openness level by deciding on how transparent or how accessible their software platform should be. Estimating the degree of openness with the six dimensions in this paper could highlight possible strategies for improving and implementing the right DOO.

7 Limitation and Future Work

This study, explorative in nature, provides a deeper understanding of some of the factors that might influence software platform providers to choose a certain DOO for their platform. As every explorative study, this work has some limitations. The first limitation is the small sample size which does not allow us to distinguish with respect to different software types. Second, since we use cross-sectional data, we can only show associations, not causality. Accordingly, future research should increase the sample size and explore other factors that might influence this decision. Therefore, we would suggest extending this framework with other theories that could contribute to gaining further insight into the DOO from a provider perspective and considering the different software types.

References

1. Tiwana, A., Konsynski, B., Bush, A.A.: Research Commentary—Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. Information Systems Research 21, 675–687 (2010)
2. Gartner, http://www.gartner.com/newsroom/id/2592315
3. Cusumano, M.: Cloud computing and SaaS as new computing platforms. Communications of the ACM 53, 27–29 (2010)
4. Eisenmann, T.R., Parker, G., Van Alstyne, M.: Opening platforms: how, when and why? In: Gawer, A. (ed.) Platforms, Markets and Innovation, pp. 131–162. Edward Elgar Publishing, Cheltenham (2009)
5. Anvaari, M., Jansen, S.: Evaluating architectural openness in mobile software platforms. In: Proceedings of the ECSA 2010, pp. 85–92. ACM (2010)
6. Boudreau, K.: Open Platform Strategies and Innovation: Granting Access vs. Devolving Control. Management Science 56, 1849–1872 (2010)
7. Aberdour, M.: Achieving Quality in Open-Source Software. IEEE Software 24, 58–64 (2007)
8. Molder, J.T., van Lier, B., Jansen, S.: Clopenness of Systems: The Interwoven Nature of Ecosystems. In: Proceedings of the IWSECO 2011, pp. 52–64 (2011)
9. Hilkert, D., Benlian, A., Sarstedt, M., Hess, T.: Perceived Software Platform Openness: The Scale and its Impact on Developer Satisfaction. In: Proceedings of the ICIS 2011 (2011)
10. Bosch, J.: From software product lines to software ecosystems. In: Proceedings of the 13th International Software Product Line Conference, pp. 111–119 (2009)
11. Jansen, S., Cusumano, M.: Defining Software Ecosystems: A Survey of Software Platforms and Business Network Governance. In: Proceedings of the IWSECO 2012, pp. 40–58 (2012)
12. Cusumano, M.A., Gawer, A.: The elements of platform leadership. MIT Sloan Management Review 43, 51–58 (2002)
13. West, J.: How open is open enough?: Melding proprietary and open source platform strategies. Research Policy 32, 1259–1285 (2003)
14. Wijnen-Meijer, M., Batenburg, R.: To Open Source or not to Open Source: That's the Strategic Question. Results from a Survey Among Eight Leading Software Providers. In: Proceedings of the ECIS 2007 (2007)

15. Barney, J.: Firm Resources and Sustained Competitive Advantage. Journal of Management 17, 99–120 (1991)
16. Winter, S.G.: The Satisficing Principle in Capability Learning. Strategic Management Journal 21, 981–996 (2000)
17. Cheon, M.J., Grover, V., Teng, J.T.C.: Theoretical perspectives on the outsourcing of information systems. Journal of Information Technology 10, 209–219 (1995)
18. Chesbrough, H.W.: The Era of Open Innovation. MIT Sloan Management Review 44, 35–41 (2003)
19. Lichtenthaler, U., Lichtenthaler, E.: A Capability-Based Framework for Open Innovation: Complementing Absorptive Capacity. Journal of Management Studies 46, 1315–1338 (2009)
20. Zahra, S.A., George, G.: Absorptive Capacity: A Review, Reconceptualization, and Extension. Academy of Management Review 27, 185–203 (2002)
21. West, J., Gallagher, S.: Patterns of open innovation in open source software. In: Chesbrough, H.W., Vanhaverbeke, W., West, J. (eds.) Open Innovation: Researching a New Paradigm, pp. 82–106. Oxford University Press, Oxford (2006)
22. Hanssen, G.K.: Opening up software product line engineering. In: Proceedings of the ICSE Workshop 2010, pp. 1–7. ACM (2010)
23. Ireland, R.D., Hitt, M.A., Vaidyanath, D.: Alliance Management as a Source of Competitive Advantage. Journal of Management 28, 413–446 (2002)
24. Buxmann, P., Diefenbach, H., Hess, T.: The Software Industry: Economic Principles, Strategies, Perspectives. Springer, Heidelberg (2012)
25. Eisenhardt, K.M.: Agency Theory: An Assessment and Review. Academy of Management Review 14, 57–74 (1989)
26. Aubert, B.A., Patry, M., Rivard, S.: A framework for information technology outsourcing risk management. SIGMIS Database 36, 9–28 (2005)
27. Bakos, J.Y., Kemerer, C.F.: Recent applications of economic theory in information Technology research. Decision Support Systems 8, 365–386 (1992)
28. Eisenhardt, K.M.: Control: Organizational and Economic Approaches. Management Science 31, 134–149 (1985)
29. Kirsch, L.J.: The Management of Complex Tasks in Organizations: Controlling the Systems Development Process. Organization Science 7, 1–21 (1996)
30. Choudhury, V., Sabherwal, R.: Portfolios of Control in Outsourced Software Development Projects. Information Systems Research 14, 291–314 (2003)
31. Xu, B., Xu, Y., Lin, Z.: Control in Open Source Software Development. In: Proceedings of AMCIS 2005, pp. 955–959 (2005)
32. Maurer, C., Tiwana, A.: Control in App Platforms: The Integration-Differentiation Paradox. In: ICIS Proceedings, Research in Progress (2013)
33. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Business Review 82, 68–81 (2004)
34. Tiwana, A.: Does technological modularity substitute for control? A study of alliance performance in software outsourcing. Strategic Management Journal 29, 769–780 (2008)
35. Cohen, W.M., Levinthal, D.A.: Absorptive Capacity: A New Perspective on Learning and Innovation. Administrative Science Quarterly 35, 128–152 (1990)
36. Hagedoorn, J., Duysters, G.: External Sources of Innovative Capabilities: The Preferences for Strategic Alliances or Mergers and Acquisitions. Journal of Management Studies 39, 167–188 (2002)
37. Wang, C.L., Ahmed, P.K.: Dynamic capabilities: A review and research agenda. International Journal of Management Reviews 9, 31–51 (2007)

38. Subramaniam, M., Youndt, M.A.: The Influence of Intellectual Capital on the Types of Innovative Capabilities. Academy of Management Journal 48, 450–463 (2005)
39. Baldwin, C.Y., Clark, K.B.: The architecture of participation: Does code architecture mitigate free riding in the open source development model? Management Science 52, 1116–1127 (2006)
40. Fitzgerald, B.: The Transformation of Open Source Software. MIS Quarterly 30, 587–598 (2006)
41. Sanchez, R.: Strategic flexibility in product competition. Strategic Management Journal 16, 135–159 (1995)
42. Hanssen, G.K.: A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. Journal of Systems and Software 85, 1455–1466 (2012)
43. Campbell, P.R.J., Ahmed, F.: A three-dimensional view of software ecosystems. In: Proceedings of the ECSA 2010, pp. 81–84. ACM (2010)
44. Rossiter, J.R.: The C-OAR-SE procedure for scale development in marketing. International Journal of Research in Marketing 19, 305–335 (2002)
45. Mayring, P.: Qualitative Inhaltsanalyse. Forum Qualitative Sozialforschung/Forum: Qualitative Social Research 1 (2000)
46. Maxwell, E.: Open standards, open source, and open innovation: Harnessing the benefits of openness. Innovations: Technology, Governance, Globalization 1, 119–176 (2006)
47. Basnet, P., Lane, M.: Informal Control in Open Source Project: An Empirical Assessment. In: Proceedings of the ACIS 2005 (2005)
48. West, J., O'Mahony, S.: The Role of Participation Architecture in Growing Sponsored Open Source Communities. Industry and Innovation 15, 145–168 (2008)
49. Cho, H., Chen, M., Chung, S.: Testing an integrative theoretical model of knowledge-sharing behavior in the context of Wikipedia. Journal of the American Society for Information Science and Technology 61, 1198–1212 (2010)
50. Alspaugh, T.A., Asuncion, H.U., Scacchi, W.: The Role of Software Licenses in Open Architecture Ecosystems. In: Proceedings of the IWSECO 2009, pp. 4–18 (2009)
51. Alam, I.: Service innovation strategy and process: a cross-national comparative analysis. International Marketing Review 22, 234–254 (2006)
52. Jensen, C., Scacchi, W.: Collaboration, Leadership, Control, and Conflict Negotiation and the Netbeans.org Open Source Software Development Community. In: Proceedings of the HICSS 2005, pp. 1–10 (2005)
53. Kirsch, L.J., Sambamurthy, V., Ko, D.-G., Purvis, R.L.: Controlling Information Systems Development Projects: The View from the Client. Management Science 48, 484–498 (2002)
54. Rijsdijk, S.A., van den Ende, J.: Control Combinations in New Product Development Projects. Journal of Product Innovation Management 28, 868–880 (2011)
55. Backhaus, K., Erichson, B., Plinke, W., Weiber, R.: Multivariate Analysemethoden: Eine Anwendungsorientierte Einführung. Springer, Berlin (2008)

Analytical Open Innovation for Value-Optimized Service Portfolio Planning

Maleknaz Nayebi and Guenther Ruhe

University of Calgary
Software Engineering Decision Support Laboratory,
University of Calgary, 2500 University Drive NW, Calgary, Alberta T2N 1N4, Canada
{mnayebi,ruhe}@ucalgary.ca

Abstract. Service portfolio planning is the process of designing collections of services and deciding on their provision. The problem is highly information and decision centric. In this paper, we present a solution approach called Analytical Open Innovation (AOI). Open innovation facilitates comprehensive crowdsourcing and social media information analysis. In our proposed approach (AOI), open innovation is utilized for the elicitation of service needs, the definition of quality provision levels for each of the services, and detection of service dependencies and cost and value synergies. As the result of a rigorous optimization process, diversified and resource-optimized service portfolios are created. As a proof of concept, the proposed approach is illustrated via a case study project using Over the Top TV (OTT) services to be offered at different levels of quality over four quarters of a year.

Keywords: Service portfolio planning, Open Innovation, Crowdsourcing, Morphological Analysis, Optimization, Case study.

1 Introduction

Services are "economic activities offered by one party to another, most commonly employing time-based performances to bring about desired results in recipients themselves, or in objects or other assets for which purchasers have responsibility" [1]. As studied in service science, management and engineering, composition of services has a broad range of applications ranging from telecommunication to health care and financial services. While services are applicable for a wide range of domains, we consider their application in IT services.

Designing service portfolios is a highly decision-centric and information-intensive process mentioned to address when? What and How good?. Service portfolio planning is increasingly under pressure to adapt to the radically changing business conditions. As a consequence, the related decision processes also need to be adapted. The reactive mode of operation needs to be replaced by real-time or even pro-active decisions. The information these decisions are relying upon needs to be highly up-to-date and comprehensive.

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 273–288, 2014.

The paradigm shift from reactive to real-time and even pro-active decision-making is of key importance for the development of services that are designed to meet market needs. Incrementally building and deploying services that rely on deep customer insight and real-time feedback is expected to create services with a higher customer hit rate that can be developed more rapidly [2].

Rometty stated in [3] that "responding to change for gaining competitive advantage in the era of smart decisions will be based not on *gut instinct*, but on predictive analytics". The paradigm of "Open Innovation*"* emphasizes on the range of opportunities that are available to get access to distributed knowledge and information. Open innovation includes methods such as crowdsourcing that has been successfully used to develop new products. It has been proven as a beneficial approach for companies and users [4] as it integrates internal and external ideas and paths to market at the same level of importance [5]. It also makes use of distributed talent, knowledge and ideas in innovation processes [6]. We are proposing *Analytical Open Innovations (AOI)* to emphasize the role and importance of analytical methods in conjunction with the data and information retrieved from open innovation.

Service portfolio design consists of several decision points. The poor systematic process and deficits in the amount and quality of data needed to make decisions, limits the design of service portfolios. Difficulty in finding and formulating the objectives and constraints gives the design of portfolios a wicked flavor. We claim that AOI is an approach to at least reduce these difficulties by not limiting knowledge elicitation and information retrieval to internal sources only. We will study Advanced Service Portfolio Planning (ASPP) as the process that relies on both internal and external information sources for (i) elicitation of service needs, (ii) defining services and their different quality levels, (iii) detection of service dependencies as well as cost and value synergies, and (iv) selection of services and their quality level to be offered in consideration of their release time

The *main contribution of this paper* is proposing a systematic method called *AOI* which supports the different steps of the advance service portfolio planning (ASPP) process. The approach is defined and justified. While AOI is an extendible platform, in this paper we focus on crowdsourcing, morphological analysis, and optimization of service portfolios. For illustrative purposes and as a proof-of-concept, we present the proposed approach by a case study on Over the Top TV (OTT) services. These OTT services, are supposed to be offered at different levels of quality over a period of four quarters of a year.

The rest of the paper is organized as follows. In the next section, we will discuss background methodology in terms of quality driven service planning, open innovation, wicked service decision making and related information needs. In Section 3, we describe casual and formal modeling of the problem domain. The solution approach is described in Section 4 and a case study will be described in Section 5. Finally, we will provide a summary of our findings and an outlook to future research.

2 Background and Related Work

2.1 Quality Driven Portfolio Planning

Since the services to be offered are still intangible, determining the demand for them is difficult where in turn this makes the service production and consumption complex. This process needs to be attuned to the needs of consumers to meet their demands successfully. A key part of this process is responding to existing and emerging consumer trends even if they are conflicting.

A poor set of quality attribute requirements causes failure in many software projects which reveals the importance of finding the right balance between the qualities offered in a set [7]. Finding the right portfolio of service qualities leads producers into more successful software products and services. Agreement of stakeholder and producer among quality requirements in domain of software projects is used as the response to contract-oriented specification compliance and service-oriented customer satisfaction [8]. Quality Attribute Risk and Conflict Consultant (QARCC) with the focus on quality attribute tradeoffs is proposed in [9] as a groupware support system based on negotiated win conditions in software domain.

Karlsson & Ryan in [10] interpret quality and its potential to achieve better customer satisfaction of the overall portfolio. By this, quality, cost and value are the dimensions which are used to prioritize functional requirements [11], and in the same manner, services.

QUPER [11] is designed to model the good enough quality in the markets considering the competitors. This provides the reason for offering a particular level of quality and facilitates decision making for release plans. Although quality requirements are of major importance in market-driven release planning [11] but several challenges in defining requirements, lowers the quality of these plans [12].

2.2 Information Needs for Release and Service Decisions

Different types of uncertainty make release decisions hard. The individual and collective value of services is hard to predict as the value depends on the number of factors which themselves are dynamically changing (e.g., competition, market trends, user acceptance). One way to mitigate these risks is to perform a retrospective analysis on existing services and based on semantic similarity, the result of analysis can be utilized to predict the value of the future services.

Historical attempts for creating products and services for answering the market's needs changed user's role and market focus. This attempt further continued by innovating products and services for having more market share and transformed into innovating with customers to create real time value in decision making process inside organizations. Aligned with this trend, open innovation helps product managers to achieve customer insight by using web 2.0 functionality. As software projects grow and become more complex and having more stakeholder across geographical and organizational boundaries, project managers increasingly rely on open discussion forums to elicit requirements [13] this approach leads us to use open innovation as the rich source for providing data for the project.

The concept of a wicked planning problem was introduced by Rittel and Webber [14]. Planning and design problems are considered to belong into this category. Wickedness, among others, refers to the difficulty to explicitly formulate the problem. To mitigate the challenge of wickedness, casual modeling is used to formalize judgmental process [15]. Release planning is considered as a semi-wicked problem and tackled with several approaches. Ngo-The and Ruhe in [16] applied an evolutionary modelling and problem solving approach for mitigating wickedness [16]. A dialogue and explanation based approach was designed in [17]. Closest to this research, a stakeholder-centric approach was proposed in [18] by Heikkilae et al which is emphasizing on the need to broaden the scope of information input at the different stages of the whole planning process [18]. The former approach would be called closed innovation with respect to what we are proposing in this paper. Besides that, we are considering service portfolios, by focusing on the validity of the solutions and by performing morphological analysis (see Section 3.1) upfront. Furthermore, we are able to utilize value and effort synergies as part of the final optimization process.

2.3 Open Innovation

Close provider and consumer relation in service firms puts a high premium on overall visibility and accountability [19]. Open innovation as a cheap and low risk problem solving approach relies on knowledge exchange with outside of company as a competitive advantage. Different forms of open innovation; crowd source, open source and outsource [6] facilitate the provider and consumer interactions. Chanal [20] defines open innovation as a paradigm which assumes that firms can and should use both external and internal ideas and paths to market.

Open sourcing is generally enabled by a web-based innovation platform [6] and will result in a product that is increasingly better, developed collectively and democratically [21]. Crowdsourcing as a type of open innovation is often enabled by the web [6] which is a creative mode of user interactivity [21]. Crowed wisdom in all of these approaches aggregates a collection of complementary data and information. Crowdsourcing is identified with other web 2.0 technologies [22]. A large chunk of the Web is about data and services. Consequently, we expect crowdsourcing to build structured databases and structured services (web services with formalized input and output receive increasing attention these days.) [23]. The idea of using large pool of problem solvers has been discussed in different contexts [6]. Further, instead of simply collaborating with a selected set of known external parties, firms are innovating by using ‘‘crowdsourcing’’ [24]

Currently, several open innovation platforms and services are available in different scopes [25] [26] but none of them are specialized in the domain of software service/product lifecycle.

3 Modeling and Problem Statement

Wicked and semi-wicked problems could be mapped into non-quantified variables and ranges of conditions. Similarly ranges of conditions can be synthesized into well-defined configurations, which represent the "solution spaces" [15]. Casual modeling is a non-quantifiable model, which is an alternative to formal methods. Casual modelling can formulize judgmental process. Solutions coming out of casual modeling could be employed as alternatives by various formal approaches.

Casual Modeling is employed in the initial, problem formulation phase of the modelling process and prior to formal models. Morphological analysis (MA) as an instance of casual modeling [27] is the study of form or structure by identifying the multiple dimensions comprising any system, and can support multi-criteria decision analysis.

3.1 Morphological Analysis

MA is a method for identifying, structuring and investigating the total set of possible relationships or "configurations" contained in a given multidimensional problem complex. MA allows small groups of subject specialists to define, link and internally evaluate the parameters of complex problem spaces as well as creating a solution space and a flexible inference model [15] [28]. MA has been applied successfully in strategic planning and decision support in various domains [27]. MA provides an "if-then" laboratory within which drivers, and certain conditions can be assumed and range of associated solutions found in order to test various inputs against possible outputs [29]. The results of a morphological model can provide input for the development of other models.

Generally, MA is intended to broaden the space of alternatives by systematic search for combination of different dimensions of a wicked or semi-wicked problem and narrow them down through the results. The result of MA is called a *morphological field*. Morphological field describes the total problem complex.

MA as a fundamental scientific method of alternating between an analysis and a synthesis phase, consisting of the following steps [29]:

Analysis phase

1. Extracting dimensions, parameters or variables, which define the essential nature of the problem complex or scenario.
2. Define a range of relevant, discrete values or conditions, for each variable.

Synthesize phase

3. Assess the internal consistency of all pairs of variable conditions
4. Synthesize an internally consistent outcome space.
5. Iterate the process if necessary.

In what follows, we provide the key concept and notation from MA which is needed to understand the rest of the paper.

Definition 1. A *morphological field* is a field of constructed dimensions or parameters which are the basis for a morphological model.

Definition 2. *Cross Consistency Assessment (CCA)* pertains to the process by which the parameter values (or parameter conditions) in the morphological field are compared with one another. This process reduces the total problem space to a smaller (internally consistent) solution space.

Definition 3. A *morphological model* is a morphological field with its parameters assessed and linked through a CCA [15].

Definition 4. A *configuration* is one parameter value or condition, displayed from each of the parameters in a morphological model.

In the context of this paper, MA will be used to elicit services based on preferences of potential customers. Fig 1-(a) shows the morphological box which forms by mining the collected data from crowd. Each column represents one service and cells of each column present values or conditions of that service.

Example: In Fig 1; the values are service quality levels and S_1 will have service levels $\{V_{11}, V_{12}, V_{13}\}$ = {Basic Service $_{S1}$, Advance Service $_{S2}$, Premium Service $_{S3}$}.

S_1	S_2	S_3	…	S_n
V_{11}	V_{21}	V_{31}	…	V_{n1}
V_{12}	V_{22}	V_{32}	…	V_{n2}
⋮	⋮	⋮	⋮	⋮
V_{1m}	V_{2x}	V_{3y}	…	V_{nz}

(a)

	S_1	S_2	S_3	…	S_{n-1}
S_2					
S_3					
S_4					
…					
S_n					

(b)

Fig. 1. (a) Morphological box of a solution space (b) Internal cross consistency assessment matrix

For diving deeper into the problem and examining internal relationship between field parameters [15], Cross Consistency Assessment (CCA) analysis is performed. This analysis acts as "garbage detector" and takes contradictory value pairs out of the solution space. Three types of contradiction can be detected:

- Logical contradictories (based on the nature of involved concepts),
- Empirical constraints (highly improbable base on empirical grounds) and
- Normative constraints.

Fig 1-(b) shows the CCA matrix which could be formed separately for each criteria, feature and plan evaluation context. As the result of CCA, some contradictory configurations are eliminated.

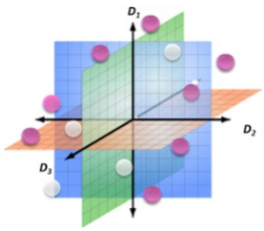

Fig. 2. Three-dimensional solution space with surviving configurations shown in violet (dark)

3.2 Problem Statement

Service release planning is targeting the delivery of optimized portfolios of the services which are most valuable to subscribers. While the problem is semi-wicked in its nature, we are providing a formal model and a subsequent analytical approach, which serves as the backbone for generating optimized portfolios. As any model, it is impossible to accommodate all details. However, adapting the principles of both open and closed innovation is intended to increase the validity of the model and the data used in it.

One key assumption of the model is that there is an existing set of potential services offered. Each of them consumes a certain amount of (fixed) upfront cost. Each service is assumed to have different levels of quality and functionality. For simplicity reasons, we call them Basic (B), Advanced (A) and Premium (P). The definition of the respective content is service and context specific. The whole approach presented later on, is applicable to more general cases as well (other types of classification).

S(n,l) $\in$ SERVICE presents a service in which n is the service number and l is the quality level. These services are planned in K release(s) where $k = 1\ldots n$ and *Release weight* $\in \{0,1,\ldots 9\}$.

A release plan is characterized as below:

$$x(n,l) =: \begin{cases} k & if\ S(n,l)\ is\ offered\ at\ relase\ k \\ 0 & other\ wise \end{cases}$$

Technological constraints, value synergy and cost synergy are modeled as forms of service dependencies. First, the set CSD of coupling dependencies are presented by set of coupled services based on the definition:

$$x(n_1,l_1) = x(n_2,l_2)\ for\ all\ pairs\ of$$
$$coupled\ services\ (S(n_1,l_1),\ S(n_2,l_2)) \in CSD$$

Similarly, the set PSD of precedence dependencies is defined by:

$$x(n_1,l_1) \bullet x(n_2,l_2)\ for\ all\ pairs\ of$$
$$precedence\ services\ (S(n_1,l_1),\ S(n_2,l_2)) \in PSD$$

Third, NAND dependency is defined. As described in Section 3.1, certain services and their related instances are not compatible with each other and do not make sense to be offered in conjunction. Detection of these incompatibilities is a complex problem itself, and we have used MA/CCA analysis to find them. NAND indicates that:

$$x(n_1,l_1)\ NAND\ x(n_2,l_2)\ if\ and\ only\ if\ services\ (S(n_1,l_1),\ S(n_2,l_2))$$
$$cannot\ be\ offered\ both$$

Each service to be provided causes certain amount of (fixed) cost. Our advanced service portfolio planning problem ASPP, assumes capacities (budgets) for the different time periods (e.g., quarters of a year) for planning. The budget related to release k is formulated as *budget(k)*.

$$\sum_{n,l:\, x(n,l)=k} s\_cost(n,l).x(n,l) \leq budget\ (k)$$

Offering a service at the basic and advanced level is requesting less cost than the sum of the individual services. We call this *cost synergies,* which is formalized as:

If $S(n_1,l_1)$ and $S(n_2,l_2) \in SCS$ then $S(n_1,l_1)$ becomes p% less expensive if offered not earlier than $S(n_2,l_2)$

Similarly, from a value perspective, the combination of certain services will increase the attractiveness to the user that is indicated with the relation to customer's ideas as below:

$$s\_value\ (n\ ,\ l) = \frac{\sum Prio(n,l,s).impotance(s)}{\sum_{s=1\ldots S} importance(s)}$$

In this case, the combination of different services is creating higher value than the individual service values. We call these the set of *value synergies (SVS)*:

If ItemSet = $\{S(n_1,l_1), \ldots, S(n_y,l_y)\} \subseteq SVS$ then:
Sum value of ItemSet is increased by Factor% if none of these items is postponed.

Similarly, the combination of different services is creating less cost than the individual service values. We call these the set of *cost synergies (SCS)*:

If ItemSet = $\{S(n_1,l_1), \ldots, S(n_y,l_y)\} \subseteq SCS$ then:
Sum cost of ItemSet is decreased by Factor% if none of these items is postponed.

Our overall problem called *ASPP* looks for service portfolios to be implemented and offered that are most attractive to users. This requires a proper understanding of user's needs. In addition, knowledge about the attractiveness of the service level in relation to the subscription fee requested form the provider is needed. We apply crowdsourcing (open innovation) for service needs elicitation and (closed innovation) stakeholder evaluation of proposed services in terms of their utility (willingness to pay for them at the level defined).

Problem ASSP: For a set of candidate services SERVICE, which includes different types of services and different types of quality and functionality at which they are offered, find a service portfolio servp* which is of maximum overall utility from the perspective of all the stakeholder inputs received as:

$$Utility = \sum_{n,l:x(n,l)\neq 0} s\_value(n,l).weight\ (x(n,l))$$
$$Utility \bullet \boldsymbol{Max}$$

The final portfolio servp* is expected to fulfill all the cost constraints and it does not contain any incompatible pairs of services delivered. Likely, it will utilize cost and value synergies between subsets of individual services.

4 Service Portfolio Planning Approach AOI

The overall AOI architecture (as illustrated in Fig 3) is designed with the aim of involving users in the process of innovation and in the development of products along

with stakeholders as the traditional source of service decisions, and then tailoring customer needs towards the organizational constraints.

The proposed AOI approach consists of three main platforms:

1- ReleasePlanner™: This pivotal platform provides a proven [30,18,31] functionality to facilitate voting and prioritization as well as generating optimized plans. The "Presentation & collaboration" component represents process outcomes to the organization and stakeholders and is responsible to initiate the work of other platforms. The "Optimization component" is responsible for the computation of optimized and diversified alternative release plans based on specialized integer programming and the special structure of the problem. The "Analysis & decision component" defines alternative features and plans with their resource consumption and degree of stakeholder excitement.

2- Open Innovation Platform: With the aim of user involvement, crowdsourcing is used. The crowdsourcing platform provides the crowd for answering questions and for facilitating control and verifying their works and task distribution. By now, Amazon Mechanical Turk [32] service, as a micro-task market, is employed. In addition, this platform, in collaboration with Very Best Choice™, maintains contact with the crowd. This software is used to manage collaboration between systems and to control the representation of feedback and to provide in house collaboration with the crowd. VBC light is a lightweight decision support system designed to facilitate proper priority decisions [33]. Moreover, text mining platform is used along with other platforms to enable automatic understanding of the crowd's response to generate meaningful data.

3- Data Analytics Platform: This platform consists of modules which are adapted with three technology pillars employed in a successful analytics platform [34], and is aligned with MA. MA is used to interpret quality in conjunction with its potential to achieve better customer satisfaction in the overall portfolio. During data extraction and creation, several separate silos of data are delivered as input to the data analytics platform. By utilizing MA approach in the data analytics platform, this data is meaningfully interpreted and classified. The large-scale computing module is evaluating large data sets of generated data as well as detecting inconsistent pairs. The analysis component is providing the results of analysis on solution space. This component consists of SSS (performs structuring of the solution space), CCA (performs cross consistent assessment), and PSS (performs structuring of problem space) modules. Along with these two components, a visualization component is needed to present the results and facilitate obtaining insights.

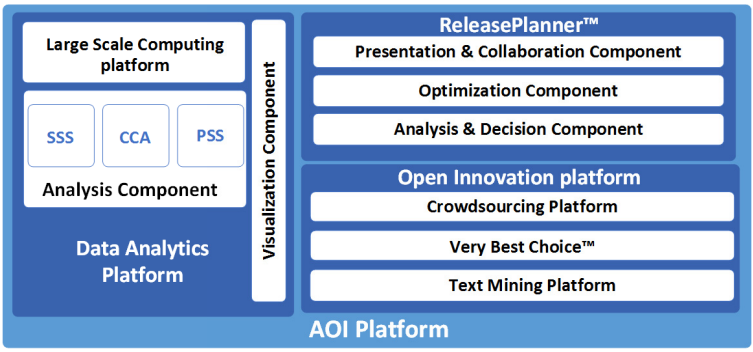

Fig. 3. Proposed AOI architecture

5 Illustrative Case Study: Planning for OTT Services

As a proof-of-concept and in order to illustrate our proposed solution approach, a case study project on *Over the Top TV (OTT)* services [35] [36] is presented. Therein, AOI is used for service elicitation (using open innovation), service quality-value elicitation (stakeholder centric/close innovation) and plan evaluation (close innovation). The OTT project encompasses one planning criteria called *willingness to pay* which is the concrete instance of the more general utility function used in Section 3.2. In order to extract features and desirable levels of functionality, several tasks such as:

- *What are the features you are looking for in an OTT service?*
- *Which level of quality (degree of functionalities) do you expect in OTT services?*

were submitted to Amazon Mechanical Turk® (www.mturk.com). Services and their related quality levels were extracted by applying text-mining techniques (see Fig 4.). AOI organizes the results in a morphological box as discussed in Section 3.1.

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Online video games	**VoIP**	**Social Network Access**	**Video on Demand (VOD)**	**Parental Control**	**Content search**	**File Sharing**	**Time Shift**	**Internet and Data**	**Multi-Screen**
Basic	Basic	Basic	Basic	Basic	Basic	Basic	Basic	Basic	Basic
Advance	Advance	Advance	Advance	Advance	Advance	Advance	Advance	Advance	Advance
Premium	Premium	Premium	Premium	Premium	Premium	Premium	Premium	Premium	Premium

Fig. 4. Morphological box of case study services

In order to detect inconsistencies as well as extracting dependencies between quality levels, Fig 4. Was further analyzed and results are shown in the CCA matrix in Fig 5.

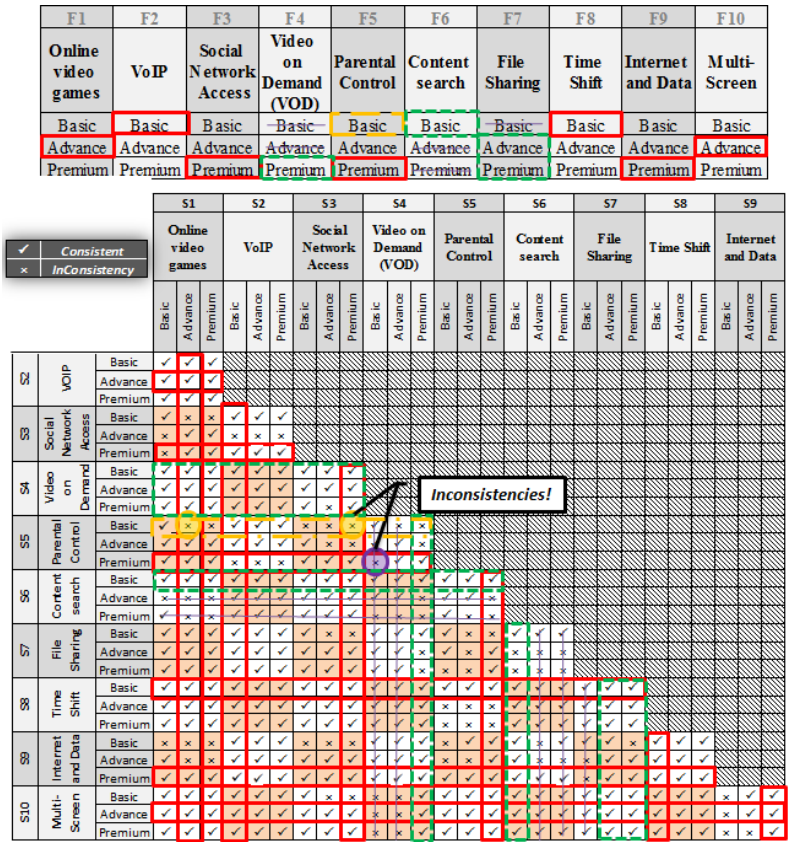

Fig. 5. CCA analysis of services

The inconsistency between premium quality levels of social network service and basic quality level of parental control service is shown in Fig 5. Such inconsistencies are taken as input for the subsequent portfolio optimization process [37].

Example: *x (Social Network Access, Premium) NAND x (Parental Control, Basic) means that x (Social Network Access, Premium) = 0 ‖ x (Parental Control, Basic) =0*

In addition to inconsistency analysis, cost and value synergy relationships between services offered in conjunction are also taken as input for portfolio optimization.

Example: *S(Multi-Screen ,Premium) service becomes 30% less expensive if offered no earlier than S(Multi-Screen, Basic)*

Example: *The sum value of {S(Online Video Gaming, Premium) , S(Social Network , Premium) , S(Parental Control, Premium)} is increased by 25% if these items are all offered in the same release*

ID	Features	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
1	Online video games Basic	5	5	5	5	5
2	Online video games Advanced	5	5	5	5	5
3	Online video games Premium	5	5	5	5	5
4	Social Network Access Basic	2	2	2	2	2
5	Social Network Access Advanced	2	2	1	5	2
6	Social Network Access Premium	3	3	3	3	3
7	Parental Control Basic	5	4	4	4	4
8	Parental Control Advanced	5	5	5	4	4
9	Parental Control Premium	2	5	2	2	2
10	File Sharing Basic	4	4	4	4	4
11	File Sharing Advanced	2	2	2	2	5
12	File Sharing Premium	5	5	5	5	5
13	Internet and Data Basic	4	4	4	4	4
14	Internet and Data Advanced	4	4	4	4	4
15	Internet and Data Premium	3	3	3	3	3
16	VoIP Basic	1	2	1	2	1
17	VoIP Advanced	2	2	2	2	2
18	VoIP Premium	3	3	3	3	5
19	Video on Demand (VOD) Basic	5	4	4	5	5
20	Video on Demand (VOD) Advanced	3	3	3	3	3
21	Video on Demand (VOD) Premium	1	1	1	1	1
22	Content search Basic	1	1	1	1	1
23	Content search Advanced	1	1	2	1	1
24	Content search Premium	2	2	2	1	2
25	Time Shift Basic	5	5	5	5	5
26	Time Shift Advanced	1	1	5	2	1
27	Time Shift Premium	4	4	4	3	3
28	Multi-Screen Basic	5	5	5	5	2
29	Multi-Screen Advanced	1	1	1	1	1
30	Multi-Screen Premium	1	1	1	1	1

Fig. 6. Portfolios generated incorporating all types of dependencies and synergies

Having stakeholders' priorities (taken from the range of {1...9}) on the service levels related to the willingness to pay, ReleasePlanner™ optimizes results and generates diversified portfolio plans. As it is shown in Fig 6 services are assigned to release numbers 1... 4. Services which are tagged with release number 5 are postponed.

Criteria for Planning	Explanation	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
9-Willingness to pay	Degree of optimality	100.0%	100.0%	94.3%	93.0%	92.7%
	(Stakeholder feature points)	(29627)	(28276)	(28022)	(27651)	(27562)

(a)

Criteria for Planning	Explanation	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
9-Willingness to pay	Degree of optimality	100.0%	100.0%	99.9%	99.9%	99.8%
	(Stakeholder feature points)	(22488)	(22477)	(22458)	(22455)	(22448)

(b)

Fig. 7. (a) Level of optimality of extracted service portfolios with synergies (b) Level of optimality of extracted service portfolios without synergies

Considering cost and value synergies, changes the sequence of services in each release. The cost and value synergies for a set of candidate services, not only changes

offered service portfolios but also makes tangible improvement in portfolio value as it is presented in terms of stakeholders' feature points in Fig 7. Higher stakeholder feature points, in companion with better degree of optimality are interpreted as higher customer satisfaction.

The final portfolio servp* fulfills all the given dependencies and cost constraints and it does not contain any incompatible pairs of services delivered. The process followed in this case study is presented in Fig 8, which is utilizing the AOI architecture given in Fig 3.

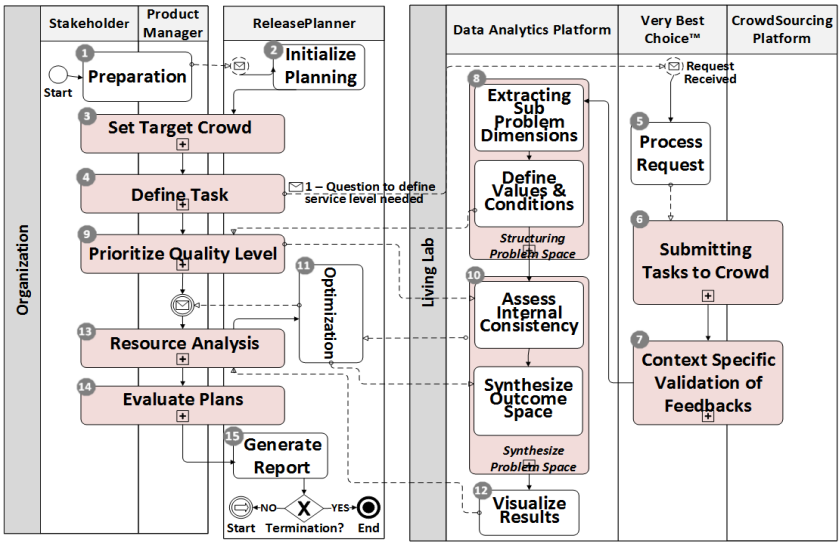

Fig. 8. Process performed for the case study

The four dimensions of the ASPP problem are covered in the AOI process as below:

(i) Elicitation of service needs: Steps 1 to 7 in Fig 8 are designed to elicit services. In order to extract customer's preferences, the AOI is using the crowdsourcing approach and the open innovation platform is designed to facilitate this procedure. With the aim of utilizing user involvement, a task is registered to the crowd. Amazon Mechanical Turk serves as a micro-task market and is used to establish the collaboration with crowd. In order to manage collaboration between systems and provide in house collaboration with crowd, the Very Best Choice™ system is used. VBC light is a lightweight decision support system designed to facilitate proper priority decision making.

(ii) Defining services and their different levels of quality at which they should be offered: Steps 8 and 9 in Fig 8 are designed to extract services and their related quality levels. Extracting services and shaping the morphological box is an attempt to structure the release planning problem space. A Well-defined set of stakeholders evaluate the individual service offerings and ReleasePlanner™ facilitates voting and prioritization.

(iii) Detection of service dependencies as well as cost and value synergies: Steps 10 in the process (Fig 8.) are designed to detect service dependencies as well as extract cost and value synergies. Assessing internal consistency with the use of CCA analysis is performed by the analysis & decision component in order to detect inconsistent pairs. The results of this analysis on the solution space are visualized and prepared to be presented to the stakeholders.

(iv) Selection of services to be offered: Steps 11 to 15 of the process (Fig 8.) are designed to extract portfolios and present the results to stakeholders. The data and information received from the analytical platform will be interpreted and categorized and then sent back to ReleasePlanner™ for performing the optimization. This system is responsible for computation of optimized and diversified alternative release plans. Optimization algorithms are based on specialized integer programming exploiting the special structure of the problem. Therein, the results are utilized from [37]. This allows accommodating more complex constraints rather than just coupling and precedence. The additional computational complexity of this more general class of planning problems is addressed by a two-staged solution procedure combining the subsequent usage of specialized integer programming and constraint programming.

6 Conclusion and Future Research

Advanced service portfolio planning is a cognitively and computationally complex problem. We have proposed a new approach for better understanding, formulating and algorithmically solving the problem, which so far has often been addressed in an ad hoc fashion. However, with the increasing demand for continuous value delivery, a more systematic method is needed, one with the capability to offer optimized service portfolios in close to real-time.

The Analytical Open Innovation (AOI) approach described in this paper is designed to address this need. AOI combines existing methods and techniques such as crowdsourcing, morphological analysis, data mining and optimization in a new way to solve this semi-wicked problem. Enlarging and improving the domain of input for data and information used in planning, we have a better chance to address *the right problem*. From applying advanced optimization techniques, we are able to include all types of dependencies between services, as well as cost and value synergies. As a result of the process, a set of diversified service portfolio alternatives is offered. Synergies result of whole portfolio costing less than when services offered individually. In the same way considering value synergies will result in higher portfolio values.

There is sufficient room for future research in this area. A more comprehensive empirical analysis is needed to demonstrate the usefulness of this approach in the domain of software. Morphological analysis can be utilized more broadly for investigating decision making criteria, services and their relation, as well as evaluation of the entire plan. At the final stage of the process, open innovation should allow continuous evaluation by further qualifying the decision-making via scenario-

playing and other forms of varying other project parameters. Finally, the scalability of the entire approach needs to be investigated to assess its effectiveness and efficiency.

Acknowledgement. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, NSERC Discovery Grant 250343-12.

References

1. Lovelock, C., Wirtz, J.: Services Marketing: People, Technology, Strategy. Prentice Hall, Upper Saddle River (2007)
2. Strategic Research Agenda Need for Speed (2013), http://www.digile.fi/Services/researchprograms/futureprograms
3. Rometty, V.M.: A Conversation with Ginni Rometty (2013), http://www.cfr.org/technology-and-science/conversation-ginni-rometty/p30181
4. Enkel, E., Gassmann, O., Chesbrough, H.: Open R&D and open innovation: exploring the phenomenon. R and D Management 39(4), 311–316 (2009)
5. Chesbrough, H.: Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business Press, Boston (2003)
6. Marjanovic, S.F., Joanna, C.: Crowdsourcing based business models: In search of evidence for innovation 2.0. Science and Public Policy 39(3), 318–332 (2012)
7. Boehm, B., In, H.: Identifiing Quality-Requirement Conflicts. Software IEEE 13(2), 25–35 (1996)
8. In, H., Boehm, B., Rodgers, T., Deutsch, M.: Applying WinWin to Quality Requirements: A Case Study. In: 23rd International Conference on Software Engineering, Toronto, Canada, pp. 555–564 (2001)
9. Boehm, B., In, H.: Aids for Identifying Conflicts Among Quality Requirements. IEEE Software 13(2), 25–35 (1996)
10. Karlsson, J., Ryan, K.: A cost-value approach for prioritizing requirements. IEEE Software 14(5), 67–74 (1997)
11. Svensson, R.B., Olsson, T., Regnell, B.: Introducing Support for Release Planning of Quality Requirements – An Industrial Evaluation of the QUPER Model. Paper presented at the 2nd International Workshop on Software Product Management, ISWPM 2008, Barcelona, Spain (2008)
12. Karlsson, L., Dahlstedt, Å.G., Regnell, B., Nattoch Dag, J., Persson, A.: Requirements engineering challenges in market-driven software development – An interview study with practitioners. Information and Software Technology 49(6), 588–604 (2009)
13. Cleland-Huang, J., Dumitru, H., Duan, C., Castro-Herrera, C.: Automated Support for Managing Feature Requests in Open Forums. C. ACM 52(10), 68–74 (2009)
14. Rittel, H.W., Webber, M.M.: Planning Problems Are Wicked Problems. Developments in Design Methodology, 155–169 (1973)
15. Ritchey, T.: Wicked Problems–Social Messes: Decision Support Modelling with Morphological Analysis. Risk, Governance and Society, vol. 17. Springer, Heidelberg (2011)
16. Ngo-The, A., Ruhe, G.: A systematic approach for solving the wicked problem of software release planning. Soft Computing 12(1), 95–108 (2008)

17. Du, G., Richter, M.M., Ruhe, G.: An explanation oriented dialogue approach for solving wicked planning problems. In: AAAI Fall Symposium (2005)
18. Heikkilae, V., Jadallah, A., Rautiainen, K., Ruhe, G.: Rigorous support for flexible planning of product releases - A stakeholder-centric approach and its initial evaluation. In: HICSS, Hawaii (2010)
19. Carman, J.M., Langeard, E.: Growth Strategies for Service Firms. Strategic Management Journal 1(1), 7–22 (1980)
20. Chanal, V.C., Marie-Laurence: How to invent a new business model based on crowdsourcing the crowdspirit. In: Conférence de l'Association Internationale de Management Stratégique, pp. 1–27 (2008)
21. Brabham, D.C.: Crowdsourcing as a model for problem solving: An introduction and cases. Convergence 14(1), 75–90 (2008)
22. Estelle´s Arolas, E.: Towards an integrated crowdsourcing definition. Journal of Information Science 38(2), 189–200 (2012)
23. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the World-Wide Web. Communications of the ACM 54(4), 86–96 (2011)
24. Majchrzak, A., Malhotra, A.: Towards an information systems perspective and research agenda on crowdsourcing for innovation. Journal of Strategic Information Systems 20(4), 257–268 (2013)
25. Gregory, D.S., Onook, O., Rajiv, K.: Rules of Crowdsourcing: Models, Issues, and Systems of Control. Information Systems Management 30(1), 2–20 (2013)
26. Ridder, P.D.: Open Innovation & Crowdsourcing Examples (2013), http://www.boardofinnovation.com/list-open-innovation-crowdsourcing-examples/
27. Ritchey, T.: Wicked Problems–Social Messes: Decision Support Modelling with Morphological Analysis. Risk, Governance and Society, vol. 17. Springer, Heidelberg (2011)
28. Ritchey, T.: Problem structuring using computer-aided morphological analysis. Journal of the Operational Research Society 57(7), 792–801 (2006)
29. Ritchey, T., Stenström, M., Eriksson, H.: Using Morphological Analysis for evaluating Preparedness for Accidents Involving Hazardous Materials. In: Proceedings of the 4th LACDE Conference, Shanghai (2002)
30. Ruhe, G.: Product Release Planning: Methods, Tools and Applications. CRC Press (2010)
31. ReleasePlanner®, Expert Decisions Inc., http://www.releaseplanner.com
32. MTurk(2013), https://www.mturk.com/mturk/
33. Very Best Choice light, Expert Decisions Inc., http://edi.lite.verybestchoice.com:3000/
34. Zhang, D., Han, S., Dang, Y., Lou, J., Zhang, H., Xie, T.: Software Analytics in Practice. IEEE Software 30(5) (2012)
35. De Boever, J., De Grooff, D.: Peer-to-Peer Content Distribution and Over-The-Top TV: An Analysis of Value Networks. In: Handbook of Peer-to-Peer Networking, pp. 961–983. Springer, USA (2010)
36. Montpetit, M.-J., Klym, N., Mirlacher, T.: The future of IPTV (connected, mobile, personal and social). Multimedia Tools and Applications 53(3), 519–532 (2011)
37. Przepiora, M., Karimpour, R., Ruhe, G.: A hybrid release planning method and its empirical justification. In: Proceedings ESEM 2012, pp. 115–118 (2012)

Observed Effects of Free Software on Software Development and Requirements Management

David Callele[1] and Krzysztof Wnuk[2]

[1] Experience First Design Inc.
Saskatoon, Canada
dcallele@experiencefirstdesign.com
[2] Software Engineering Lund University, Sweden
Krzysztof.Wnuk@cs.lth.se

Abstract. [Context & motivation] Free software is often declared to be a positive movement that makes technology accessible to those who might not otherwise have access.

[Problem] While the positive effects, to one degree or another, have often been discussed there has been relatively little discussion of the possibly negative effects of the free software movements. In general, these approaches have led to ever-increasing concentration of economic power in a smaller number of entities, the reduction of margins to the point where there is little economic incentive to investment and a general casino-like approach to software development: build it, then build a customer base and then try to find a way to monetize the customer base rather than the product.

[Contribution] The resulting conditioning of the customer base has led to a significant reduction in the availability of venture capital for software products and a corresponding increase in the availability of venture capital for software as a means to make the customer the product through data analytic approaches. This short paper discusses these observed effects of free software on the software economy and possible effects on requirements engineering practice.

Keywords: free software, requirements management, negative effects, software product management.

1 Free Software

Since the early experiments at Berkeley and MIT [4] in the 1960s and the first public release of the Linux kernel source in 1991 [1], free[1] and open source software have deeply penetrated and shaped the software industry [4]. The scale of these sociological movements has far exceeded the founding expectations. Today, many large proprietary software products have a free and open source counterpart, some of which have been shown to deliver equal or greater customer experience than the original [2].

[1] This work uses free in the context of "without fee or consideration". This work does not address free in the context of "free as in freedom" as is often used by organizations like the Free Software Foundation, http://www.fsf.org/ .

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 289–294, 2014.

The Internet has reshaped many industries by enabling direct-to-consumer communication and distribution. We posit that the continued practice of making software and software-based services available for free could have Internet-like effects upon the practice of requirements engineering, the software industry as a whole and upon the consumers of the software themselves.

In this paper, we present and discuss our observations of the effects of free software on investments in software development and the potential ramifications for requirements engineering.

2 Related Work

For the purposes of this section, we shall assume that the participants being discussed are rational in the economic and financial sense. In other words, some form of cost-benefit analysis is performed and the necessary investment in developing software delivers a return that is acceptable to the investor. The Return On Investment (ROI) is not necessarily monetary; for example, making a donation to an open source effort is not necessarily economically rational, at least from the perspective of some observers. But, for the individual making the donation of their time and effort there exists some form of return on that investment that makes the investment worthwhile to them.

Lerner and Tirole investigated the open source movement from an economist's perspective [3, 4] and found many aspects that were "economically puzzling". They were able to identify that contributor ROI is dominated by career-based motivations and ego gratification, observing that open source projects may be more 'cool' to contributors than their routine development tasks. On the other hand, Scacchi [5] focused on understanding how the development of requirements for open source software differs from the development of requirements for commercial software, stressing that open source software initiatives do not adopt modern software and requirements engineering practices with the absence of formal requirements elicitation, analysis and specification activities. Lerner and Tirole [3] confirmed these observations, noting that expected project and product management practices were less stringent in non-commercial projects.

3 Cost of Production and Reproduction

Determining the ROI for free software requires an understanding of the cost of production and reproduction of software. We model the reproduction cost of software at zero since the incremental energy cost of transmitting a copy is very small. Typical production cost calculations are dominated by the developer time invested in the project. However, these are not the only costs that should be considered and we look more closely at two factors that are not often taken into consideration by software developers[2].

[2] The work of economists Lerner and Tirole [3, 4] does investigate some aspects of this issue.

Lost Opportunity Cost (LOC) is often ignored in cost analyses for software development. Lerner and Tirole mentions that when a programmer puts time in an open source project is usually unable to engage in another programming activity [3]. In the context of a single developer, what else could that developer have been doing with the time that they spent working on the software? Let us assume that the developer is employed by a third-party that develops commercial software (software that must be paid for by the user). Outside of the normal work week, the developer could choose to develop software that will be freely distributed when completed. Or, the developer could choose to work extra hours for their employer and the employer's product could enter the market more quickly. Which effort should the developer support? Developer time is typically a zero-sum game and the decision to work on one project means another project does not get those resources.

Lost opportunity cost models for open-source projects can be even more complex. Open source projects that are commercially focused must resolve issues of shared intellectual property, especially when the open source license requires the sharing of all modifications. Deriving a custom branch from the main open source branch to deliver significant proprietary innovations with substantial customer value can lead to unexpected maintenance and bug fixing efforts – responsibility for keeping the now proprietary codebase current with the open source code base is no longer a shared effort. Only a proper analysis can determine whether the adoption of the open source code base is financially justified for a given project.

Some models also consider LOC at the society level. For example, the developer could have been working on software for their employer rather than on free software. Given that the employer is a for-profit entity, this activity is expected to lead to greater economic activity and a probable increase in the corresponding tax base. Does the same effort devoted to free software lead to the corresponding increase in the tax base? Perhaps it leads to a reduction in costs, thereby increasing margins and delivering the same net economic benefits? Does the society-level analysis outweigh the personal analysis?

Life Cycle Cost Analysis is another analysis that is often ignored in software cost analyses. Even if the reproduction cost for a software product is zero, the replacement costs for installing a new version of the same product can be very high – particularly when data must be migrated and users must be retrained.

These few examples illustrate that free software is not free, there are non-trivial investments required to conceive, design and implement the software. These resources must come from somewhere and these resources are, of necessity, not applied to alternative projects. Determining the appropriate model for calculating the cost base is subject to interpretation and the resulting values can vary widely.

4 Dumping, the WTO, and Free Software

The cost of software production is important in the context of international trade, particularly as economies attempt to diversify into the so-called knowledge economy. International trade is generally governed by the rules of the World Trade Organization (WTO). In the realm of tangible goods, those goods that consume incremental materials in the

production or reproduction of that good, international trade is governed in part by dumping rules. Simplistically, dumping is considered a form of *predatory pricing* that occurs when a supplier in one country charges consumers in another country a price that is either below the price charged in their home country or *below the cost of production*. Accusations of predatory pricing have been made against dominant software industry players such as Microsoft[3] and Google[4]. While regulatory powers such as antitrust have been brought to bear upon specific instances, we are unaware of any large-scale consideration being given to whether dumping constraints should also apply to intangible goods such as software. In other words, is there a basis for making the argument that the practice of giving away software for free constitutes dumping?

We look at the mobile telephone industry to see an example of the possible commercial effect of a free and open-source solution, a commercial effect that dumping regulations are intended to preclude in the realm of tangible goods. The introduction of the iPhone transformed the consumer mobile device marketplace, exerting great competitive pressure on traditional handset manufacturers. It also exerted great pressure upon Google because Apple maintained control over consumer data generated by these devices. In response to this threat to its data acquisition business, Google delivered the Android operating system as a free alternative to Apples proprietary operating system. Google's decision to protect their data acquisition market through the release of a free handset operating system eliminated the significant barrier to market entry associated with handset operating system development and rendered the massive operating system investments by handset manufacturers such as Nokia and Sony-Ericsson essentially valueless. Now almost anyone could become a handset provider – all they had to do was build the hardware, the operating system comes for free. This forced handset providers to search for differentiations in hardware and form factor.

For software to be subject to dumping rule analysis, appropriate pricing models would be required. As we saw in the prior section, simple models (*e.g.* reproduction cost is zero) are unrealistic but determining the appropriate level of complexity could require significant negotiation (*e.g.* an analysis that looked only at the cost of labor would place all developed nations at a significant disadvantage in the sector).

If free software was subject to dumping rules, how would this affect the practice of requirements engineering? Would any additional factors need to be considered or would RE practitioners argue that this was outside of the scope of their responsibility? Would this not be a regulatory compliance requirement just like safety or taxation?

5 Observations on the Effects of Free

The various app marketplaces for mobile devices contain millions of apps. As the number of available apps has increased, getting the consumers attention has become increasingly difficult. Many developers now give away their apps for free in the hope

[3] http://www.theguardian.com/technology/blog/2006/jun/21/microsoftaccus

[4] https://fsfe.org/activities/policy/eu/20130729.EC.Fairsearch.letter.en.html

that they can obtain users, trusting that they will be able to derive revenue from the users once they become tied to using their apps. However, consumers have rapidly become acclimated to the concept of apps being free and there is now significant reluctance to pay for apps when they might soon become available for free – either from the original developer or from someone cloning the concept.

The ability for a second or third party to simply clone the functionality of a software application without consequence is a significant competitive market risk for the innovating developer. Investing in new product development usually requires significant resources and if someone else can just copy or clone the innovation then why invest in that development in the first place? This is a very real threat to ongoing innovation in our industry and has led to a significant reduction in the placement of investment funds in the sector except when the project can demonstrate a significant barrier to competitive market.

Significant investment is currently being placed in projects where the software facilitates the collection of unique data about the users of the software. This unique data has value to third parties such as marketers and entrepreneurs are monetizing this data to ensure their revenue stream in the presence of customer expectations for free software. Essentially, the developer becomes a developer of software that converts data about the user of the software product into the item of value rather than brings value to the customer as features or quality attributes.

In the Angel investment group of which the first author is a member, in the last three years less than 6% of all investment pitches in this sector received investment. In every case where an investment was placed, the investment hinged upon the extent to which the users were monitored and data about them gathered. Anecdotally, this result is typical across North America.

The availability of free software has, therefore, had at least two impacts on the practice of software development. It has led to a chilling effect on investment unless there is some way to reduce the threat of a competitive clone of the product. And, in response, the industry has shifted to monetizing user data instead of monetizing the product itself. This makes the software business much more dependent on legislation bodies that may allow or prohibit collecting, monitoring and monetizing user data.

As a result, it may be necessary that the practice of RE expand its scope. Traditional RE that focusses on features is still mandatory for without the right set of features there will be no users. However, RE practices may now need to include RE for user data acquisition to support monetization efforts or there will be little or no revenue to sustain operations. Further, we may even see RE efforts that focus on mechanisms for convincing the user to give up their privacy in exchange for the service.

Requirements engineering for data acquisition can be very technically and legally complex. Effective and minimally intrusive data acquisition mechanisms require significant design of experiment and data science expertise and RE efforts may require the addition of subject matter experts in these fields. Ensuring compliance with privacy legislation in all operating jurisdictions for a product or service is another significant effort. Work on privacy and RE is in its nascent stages and interested readers should investigate the RELAW series of workshops for further information. While there have been mentions of privacy regulation compliance in published work, we are

unaware of any work that focuses on how the general practice of RE may need to change to accommodate these needs.

6 Conclusions

Despite widespread adoption of free and open source software, studies that explore the negative effect of this phenomenon on the software business economy are rarely reported in the technical literature. In this paper, we present a number of observations on the consequences of free software. We identify that software is generally exempted from international trade restrictions on predatory pricing (although some antitrust and monopoly actions have been taken against the largest industry members). The advent of a marketplace that expects software to be free has had a chilling effect on investment unless there is a substantial barrier to competitive market entry. Data collection about the users of the software has become the barrier to competitive market entry, turning the users into the main source of the revenue stream.

We are unaware of any easy answers to the observed dilemmas. Perhaps we could mandate that customers have the right to opt out of this data collection by paying a fee. What would the value of an organization like Facebook or Google be if they had direct paying customers? What would happen if we applied dumping guidelines to software pricing? Would this approach result in a more equitable and privacy-preserving marketplace or would this have a significant negative effect on innovation?

In future work, we would like to further study these issues by empirically evaluating the assumptions, hypotheses and personal experiences brought forward in this paper. Once these observations are better grounded, we can return our attention to their effects on the practice of RE and potentially develop new practitioner guidance.

Acknowledgements. This work is founded by the SYNERGIES project, SNS Foundation, grant 621-2012-5354.

References

1. The description of the Linux kernel project is available at, `http://en.wikipedia.org/wiki/Linux_kernel`
2. Stamelos, I., Angelis, L., Oikonomou, A., Georgios, A., Bleris, L.: Code quality analysis in open source software development. Inf. Sys. Journal 34, 43–60 (2002)
3. Lerner, J., Tirole, J.: Some Simple Economics of Open Source. Journal of Ind. Econ. L, 197–232 (2002)
4. Lerner, J., Tirole, J.: The Economics of Technology Sharing: Open Source and Beyond. Journal of Economic Perspectives 19, 99–120 (2005)
5. Scacchi, W.: Understanding the requirements for developing open source system. IEE Proc.-Soft. 149 (2002)

The Preliminary Results from the Software Product Management State-of-Practice Survey

Andrey Maglyas[1] and Samuel A. Fricker[2]

[1] Software Engineering and Information Management,
Lappeenranta University of Technology, Finland
andrey.maglyas@lut.fi
[2] Software Engineering Research Laboratory,
Blekinge Institute of Technology, Sweden
samuel.fricker@bth.se

Abstract. Software product management (SPM) as a discipline includes many practices like product and release planning, market analysis, roadmapping, and product lifecycle management. Product management frameworks prescribe these practices but companies seldom adopt all of them. We conducted a state-of-practice survey with the aim to investigate how companies adopt SPM practices and how this practical experience fits together with the framework suggested by International Software Product Management Association (ISPMA). The results of this study showed that ISPMA SPM Framework describes core product management practices well but the impact of product management practices to the final product success remains ambiguous.

Keywords: software product management, state of practice, survey.

1 Introduction

Software product management (SPM) unites business and technical perspectives in the development of software products. SPM defined as *business management at the product, product line, or product portfolio level* [1] in a software organization [2] represents a model for strategizing, conceiving, developing, introducing, managing, and marketing new products to the market.

There are several frameworks developed to address the specific features of managing software products [2,3,4,5]. They describe the structure and content of software product management as lists of practices that should be adopted by companies. These lists include from 16 to 38 practices. Companies rarely adopt all product management practices and focus on subsets of them that bring most benefits to the business [6]. In contrast, the existing frameworks provide little guidance on how to adopt them iteratively rather than instantly [6]. Understanding and inclusion of these priorities observed in practice to frameworks would be an important step for further development of SPM education, research, and practice. The ISPMA SPM Framework v.1.1 [2] was chosen as a reference model for this study because it represents a consensus between industry and research that integrates previously known reference models.

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 295–300, 2014.

2 Background

There have been some attempts to highlight the most important practices in product management for achieving product success. For example, Kittlaus and Clough divide SPM practices into core and supporting practices at product and corporate levels [3]. Core practices are major functions in which a software product manager is involved while supporting practices are orchestrated by product managers but not directly managed. Using the same definition of core and supporting SPM practices, Maglyas et al. identified six core practices and concluded that it is reasonable to expect an expertise in these practices from every product managers while other skills may depend on the domain and type of product [7].

The results of these empirical works are not conclusive, however. Core product management practices and responsibilities of product managers vary from one study to another depending on the framework with which the assessment is done. Such heterogeneity is not a new problem, though, and has been addressed with industry standards that offer consolidation.

In order to consolidate the existing knowledge and experience in the field of software product management, the International Software Product Management Association (ISPMA) created its SPM framework [2,8].

3 Research Methodology

This study investigated product management practices with the ISPMA reference model. It aimed at understanding how SPM practices described by ISPMA fit together with SPM practices used in real life and thereby give decision-support for the adoption of SPM practices. Two research questions were defined as follows:

- RQ1: *Does the ISPMA framework reflect software product management practice?*
- RQ2: *Does practice differ between junior and senior product managers?*

A survey followed by a focus group discussion with software product management experts was selected as the main research tool.

ISPMA SPM framework v.1.1 consists of 38 practices involved into development and release of a product to the market. These practices were grouped into several questions according to the framework structure. Each question was related to one column of the framework and was formulated as follows:

Which of the following practices are/were performed with you feeling responsible for?

The first option for answers was exclusive (*not leading any XXX practice*, where XXX is the name for a group of practices in the framework). The survey was conducted using a web-service called FluidSurveys[1]. Invitations to participate in the survey were distributed using the snowballing technique [9].

[1] `http://fluidsurveys.com`

The survey was conducted for a period of six months started in October, 2012 and finished in March, 2013. Then, the gathered results were discussed with experienced product management professionals from industry and academia at the ISPMA member assembly meeting in April 7, 2013. In this meeting, additional input on how the results fit with practice was collected in the form of meeting notes.

4 Results

The survey was answered by 100 respondents. 48 responses were incomplete, five responses were test fillings, and one response was excluded from the analysis as an outlier due to its ridiculous answers. The demographic information about the respondents and companies they work for is presented in Figure 1.

Company			Product			Development		
Size (number of employees)			**Industry (application domain)**			**Time-to-market**		
<10	3	7%	Software / IT	11	24%	Less than 4.5 months	10	22%
10-49	15	33%	Medical / Health Care	7	15%	4.5 months to < 9 months	18	39%
50-249	11	24%	Banking / Finance	5	11%	9 months to < 18 months	14	30%
250-4499	12	26%	Manufacturing	3	7%	More than 18 months	3	7%
>=4500	5	11%	Media / Publishing	3	7%	I do not know and cannot estimate	1	2%
Age (years)			Government / Military	2	4%	**Release heart-beat**		
<3	13	28%	Private / Consumer	2	4%	More than 12 releases per year	3	7%
3-7	11	24%	Retail / Wholesale	2	4%	5-12 releases per year	5	11%
8-15	9	20%	Telecommunications	2	4%	3-4 releases per year	8	17%
16-39	7	15%	Construction / Contracting	1	2%	About 2 releases per year	17	37%
>=40	6	13%	Insurance	1	2%	About 1 release per year	7	15%
Location			Nonprofit Institutions	1	2%	Less than one release per year	5	11%
Netherlands	14	30%	Other	6	13%	No release so far	1	2%
Sweden	8	17%	**Product team size**					
USA	7	15%	<4	5	11%			
Finland	5	11%	4-9	14	30%			
Switzerland	5	11%	10-19	9	20%			
Russia	4	9%	20-49	9	20%			
Czech Republic	1	2%	50-249	7	15%			
Germany	1	2%	>250	2	4%			
Denmark	1	2%						

Fig. 1. Demographics of the collected data

In the survey, we asked respondents to mark product management practices that they are responsible for. In general, SPM follows some key practices but there is variation between other practices. In more than 75% of the cases, product managers were responsible for five SPM practices: positioning and product definition, business case and costing, roadmapping, release planning, product requirements engineering. In this regard, these practices represent core product management practices observed in practice. In addition, all these practices are included to the SPM framework as core practices as well.

Another set of five SPM practices (innovation management, product analysis, product lifecycle management, project requirements engineering, product launches) was observed as related to product management by more than 50% but less than 75% of the respondents. Two of these practices (product analysis

and product lifecycle management) considered as core SPM practices by ISPMA framework but in practice only some product managers take responsibilities of these practices.

In our analysis we also compared how the work of SPMs differs from the work of senior SPMs. The sample included 23 product managers and 11 senior/head product managers. The results of two-tailed difference of proportion test are presented in Figure 2. We used Holms step-down method [10] for limiting the alpha error that accumulates over the repeated application of individual statistical tests. Here an alpha error is an apparent difference, which in reality is not a difference but just represents the luck of the investigators. This approach ensures that the total error is below the initially predetermined threshold.

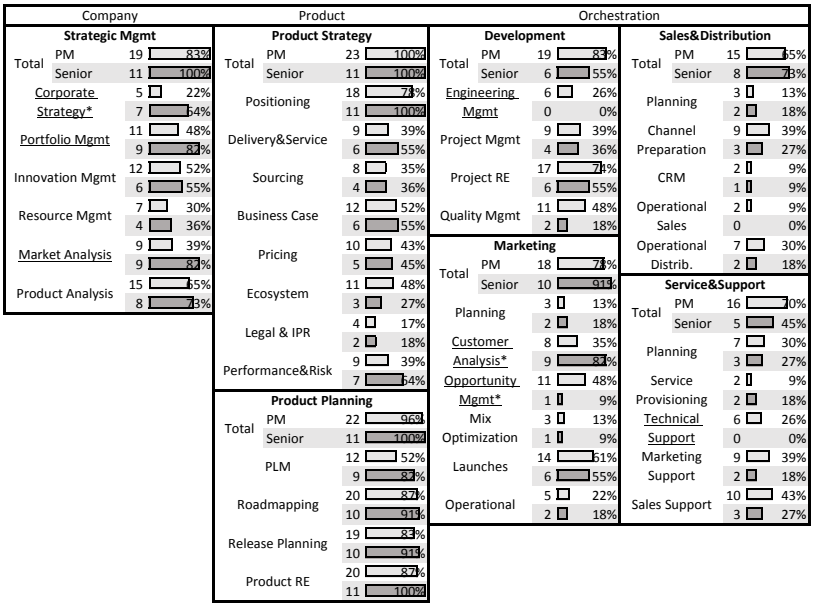

Fig. 2. Differences between software product managers and senior software product managers (two-tailed difference of proportions test, $\underline{p<0.05}$*, $\underline{p<0.1}$)

Our results revealed that non-senior SPMs tend to interface more with development and operations than senior SPMs. Senior SPMs tend to be involved in corporate strategy more than SPMs. In particular, the senior SPMs have more often an active role in the definition of corporate strategy and portfolio management. To support these decisions, they are more frequently involved in market and customer analysis than SPMs. Regardless of their seniority, product managers are heavily involved into product strategy and product planning while their involvement in marketing, sales & distribution, and service & support is limited. Only a few product managers marked these practices as being under their control.

5 Discussion

Maglyas et al. investigated core product management practices in another survey conducted worldwide and concluded that core product management practices are product analysis, roadmapping, strategic management, vision, product lifecycle management, and internal and external collaboration [7]. Product analysis, roadmapping, and product lifecycle management practices were identified as core practices in this survey as well. Strategic management is included in the ISPMA framework as a set of practices consisting of other practices and therefore cannot be directly compared. The core practice vision is included into the ISPMA framework as business case and costing. The results showed that 78% of respondents were responsible for this core practices and therefore the results fit well with the ISPMA framework and Maglyas core SPM practices. Internal and external collaboration is not included as a separate practice in the SPM framework but it is embedded to the framework structure through practices in which a product manager participates or orchestrates.

Overall, the ISPMA framework structure has several misalignments with practical experience of product managers in the software industry. Some core practices like pricing, legal and IPR management that were not often implemented by product managers as their main responsibilities represent variations in the adoption of SPM. A framework like the ISPMA SPM framework should make such differences between recommendation and practice explicit by providing rationales for the recommended infrequent practices and suggesting criteria regarding their adoption.

The analysis of responsibilities of senior software product managers and software product managers revealed that senior product managers tend to be responsible for the practices related to strategic management like corporate strategy, portfolio management, and market analysis while non-senior product managers tend to be responsible for orchestration functions like engineering management, opportunity management, and technical support.

As a unified group product managers can be seen middle managers who act as linking pins connecting the top management with the lower-level managers [11]. As an individual in this mediating position between strategic and operational levels, the product manager tends to move to senior product management position.

The main limitation of this study was the size of sample that was a result of low response rate. Increasing the sample size would help to get more statistically significant results. However, these preliminary results provide us with some insights on how product management practices are adopted in organizations and therefore can be used for generating hypotheses for new surveys with more focused questions on particular SPM practices.

The use of snowballing with a particular focus on the ISPMA network led to a non-random sample but we accepted the non-random sampling as a trade-off.

6 Conclusions

The survey results provide a general overview of how SPM is adopted in practice and how the adoption of SPM fits together with the theoretical ISPMA SPM framework that represents a consensus between industry and academia. However, due to the limited number of responses, we could not identify success-correlating practices.

The empirical validation of core product management practices described in the ISPMA SPM Framework showed that product managers are responsible for most of the suggested practices in their daily work. Leaving out the variations between different companies, the SPM Framework provides a good reference point to what product managers should be responsible for. These results are also aligned with previously identified six core product management practices [7].

Overall, the survey gives us insights to the state-of-practice in the field of software product management and contributes to the product management body of knowledge. The presented results are a basis to adapt the theoretical frameworks to real-world practice.

References

1. Haines, S.: The Product Manager's Desk Reference. McGraw-Hill (2008)
2. Fricker, S.A.: Software product management. In: Maedche, A., Botzenhardt, A., Neer, L. (eds.) Software for People. Management for Professionals, pp. 53–81. Springer, Heidelberg (2012)
3. Kittlaus, H.B., Clough, P.: Software Product Management and Pricing. Key Success Factors for Software Organizations. Springer (2009)
4. van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: Towards a reference framework for software product management, pp. 319–322 (2006)
5. Ebert, C.: Software product management. Crosstalk 22(1), 15–19 (2009)
6. Maglyas, A., Nikula, U., Smolander, K.: Comparison of software product management practices in SMEs and large enterprises, pp. 15–26 (2012)
7. Maglyas, A., Nikula, U., Smolander, K.: What do practitioners mean when they talk about product management? 261–266 (2012)
8. ISPMA: International software product management association (ISPMA) (2012)
9. Groves, R.M.: Survey methodology. Wiley-Interscience, Hoboken (2004)
10. Gordon, A.Y.: A new optimality property of the holm step-down procedure. Statistical Methodology 8(2), 129–135 (2011)
11. Floyd, S.W., Wooldridge, B.: Dinosaurs or dynamos? recognizing middle management's strategic role. Academy of Management Executive 8(4), 47–57 (1994)

Alignment Issues in Chains of Scrum Teams

Jan Vlietland[1] and Hans van Vliet[2]

[1] Search4Solutions B.V., Professional Services, Utrecht, The Netherlands
j.vlietland@Search4Solutions.nl
[2] Faculty of Sciences, Division of Mathematics and Computer Science,
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
J.C.van.Vliet@vu.nl

Abstract. IT functionality in companies is often delivered by a chain of interdependent software applications. To handle changing business demands in such chains, organizations increasingly employ agile methods such as Scrum. Usually, each Scrum team deals with a single application in the chain, necessitating alignment activities between the Scrum teams that jointly deliver IT functionality. We empirically investigate the alignment issues that occur in such chains of Scrum teams, and identify several that impact time to market. We base our observations on qualitative data from two case study environments.

Keywords: Agile, Scrum, chain of Scrum teams, priority alignment, feature time to market.

1 Introduction

Most application service providers (ASP) of large companies operating in the information intensive industries experience rapid changing business demands. To handle such changing demands, ASPs increasingly adopt Agile methods, such as Scrum. Scrum intends to mitigate delivery risk by the use of swift feedback [1].

Functionality in ASP environments however is delivered by a portfolio of interdependent applications, not by one single application. As a consequence, Scrum teams that deliver the applications likely need to align their activities. Each application in the portfolio supports a business function in the front to back business process. Together, the front to back business process delivers features, i.e. intentional distinguishing functional characteristics of the application landscape that can be used by a business user. When a feature needs to be added or changed, multiple interdependent applications need to be changed simultaneously.

Scrum teams can be mapped in different ways onto the application landscape. Some prefer to have one Scrum team for the whole front to back chain. However, the amount of involved IT staff then easily exceeds the generally agreed upon maximum size of a Scrum team. Also, changes require highly specialized skills that cannot be easily shared, and so dedicated Scrum teams exist for (business) applications. Given the interdependencies in the application chain, multiple Scrum teams then need to jointly deliver new or changed features. Such joint delivery implies that Scrum teams

C. Lassenius and K. Smolander (Eds.): ICSOB 2014, LNBIP 182, pp. 301–306, 2014.

need to cooperate. Yet, due to the nature of Scrum teams, such cooperation might not happen naturally. A Scrum team is accountable for its own application, not for the chain of interdependent applications, likely resulting in a bounded Scrum team focus rather than a delivery focus of features.

The usage of Scrum in large organizations is relatively new and academic literature in this area is scant, while there is a considerable need for answers in this area [2, 3]. In this study we empirically investigated and found four issues that exist in chains of codependent Scrum teams: (1) misalignment of backlog priorities, (2) alignment issues between teams, (3) unpredictability of committed deliveries and (4) lack of coordination in the chain.

2 Related Work

We look at the topic from two perspectives. From an Agile perspective we explain the social nature of a constellation of Scrum team, focusing on the intra- and inter-team communication issues that are relevant in our setting. As a chain of Scrum teams has similarities with supply chains we also look from a Supply Chain perspective.

Scrum teams likely has a natural intra-team focus rather than inter-team focus, as the team focuses their software development effort on a single prioritized backlog [4]. The intra-team focus is enforced by the closeness of staff within a team and their similarities (e.g. shared goal, IT application, backlog). Leffingwell [5], author of the SAFe model, argues that the product backlog should be aligned by collaborating product owners. Operational alignment between codependent Scrum teams during the software development cycle is achieved by Scrum of Scrum meetings [6].

Supply Chain Management (SCM), our second angle, is the management of interconnected network of businesses involved in the provision of product and service packages [7], which is related to a chain of Scrum teams. Workflow in supply chains is managed by means of collaboration rather than centralization. Chain oriented structures rely on collaboration, co-ordination and communication, abbreviated as 'Three-C' [8]. Also other SCM research confirms the importance of Three-C in well performing supply chains [9-11]. Given the interdependencies in the application chain, we expect that Three-C practices are also needed between codependent Scrum teams.

3 Case Study Results

We performed qualitative [12] research at multinational service providers to gain an in-depth understanding of the issues in interdependent Scrum team chains [13].

We performed two case studies at large multi-national organizations, one at the telecommunication company (9 interviews in two overlapping chains) and one at a retail bank (6 interviews in one chain). Each of the companies has a centralized IT organization with 250-1500 IT development employees who offer application services to front-office, operations and finance business functions.

We transcribed the interviews and coded them. Table 1 shows the number of codes in each main code category. The numbers indicate the grounding of the code category.

Table 1. Code grounding in each code category

Concept	Case 1	Case 2	Total
Alignment	82	15	97
Coordination	84	9	93
Prioritization	56	23	79
Predictability	30	6	36
Total	**252**	**53**	**305**

Alignment issues between Scrum teams is considered the main issue (97x grounded). We found evidence of misaligned definitions of done, sprint cycle differences (start, duration, finish), disconnected test environments and misaligned test scenarios between teams.

We identify lack of coordination as the second main issue in chains of Scrum teams (93x grounded). In particular test coordination is considered difficult, notwithstanding that we found coordination issues throughout the full sprint lifecycle.

Mismatched prioritization between teams are considered the third main issue (79x grounded). Priority setting mismatches are mainly caused by goal differences at strategic level that are cascaded to the applicable product backlogs.

The fourth issue is unpredictability during the development lifecycle in the codependent Scrum teams (36x grounded). In case one of the teams is unable to deliver its functionality that is required for a feature, the delivery of that feature is delayed. Such unpredictability risk increases with the number of codependent teams. For instance if each team delivers the necessary functionality in 9 out of 10 cases and 10 codependent teams exist, the overall predictability is $(0.9)^{10}$ being less than 35%.

Table 2 shows for each of the four main issue categories the percentage of quotes in each case study. The table shows that case study 2 mainly exhibits prioritization and alignment issues, while case study 1 mainly exhibits alignment and coordination issues.

Table 2. Cross-case analysis of the four main issues

Concept	Case 1	Case 2	Total
Alignment	33%	28%	30%
Coordination	33%	17%	25%
Prioritization	22%	43%	33%
Predictability	12%	11%	12%
Total	**100%**	**100%**	**100%**

3.1 Case Study 1: Retail Banking Front to Back Application Chain

The retail bank has a centralized IT organization with 150 Scrum teams that deliver web application services to the various business lines that deliver the services to the banking customer. The web applications are developed by front-end Scrum teams that

are clustered around different channels (e.g. internet, call center). The web applications interact with back-office applications that are clustered around financial products. The back-office applications are mainly commercial of the shelf (COTS) packages. These packages are configured in Scrum teams. The back-office packages interact with finance applications that are developed by waterfall organized IT teams. Next to the regular Scrum-roles the organization has an integrator role for the coordination of new front to back feature development.

Priority mismatches result in Front-end teams developing stories for features that cannot be delivered, because Finance teams develop high-priority non-related stories (e.g. for compliance regulation).

"The managing directors in our board have different priority settings. The managing director would like to improve his business process, while the manager of the Internet Channel needs to resolve a number of compliance issues and the manager of Marketing and Sales want to develop software for new product offering. They all think that their objective will be achieved, however without making an explicit choice in priority setting", IT development manager

Priority mismatches have several negative consequences. Longer time to market of feature delivery is one. Another consequence is concurrent integrators fighting for the highest priority in the chain, while the product owner naturally follows priority based on the strategic objectives of the managing director.

"*If the product owner does not want to cooperate in delivering interdependent functionality I do not have any influencing authority",* Integrator

Alignment issues exist in particular during front to back feature testing. Testing is a complex, labor intensive and highly interdependent exercise. During testing teams need to jointly prepare and execute front to back feature testing, requiring the alignment of the interdependent activities, definitions and terminology.

"*A lot of work needs to be redone to synchronize test data to ensure that the correct test data is available in the test environment that needs to be used in the internet channel",* Integrator

These interdependencies require extensive coordination between teams, which is performed by an integrator role, in case more than 3-4 teams are involved.

"With more than 3 to 4 codependent teams a cross-team coordination mechanism is needed between these teams. Until 3-4 teams, coordination is done by product owners.", IT manager

3.2 Case Study 2: Telecommunication Front to Back Application Chain

The telecom company has a centralized IT organization having 34 Scrum teams. The IT organization has a cluster of front-end Scrum teams which develop applications that interact with Scrum teams of Operation that interact with Scrum teams for Billing and Finance. The involved applications are interdependent, implying that a feature can only be delivered by the constellation of front to back applications. We interviewed Product Owners, Scrum Masters, IT development managers and project managers. Interviewees in this case also refer to priority conflicts between interdependent teams:

"*We are executing 15 programs concurrently…. Each program has its executive at business side which has its own profit & loss. So, who is more important when each executive want to have their features first",* IT manager

For high priority features the needed stories in the interdependent teams are placed as highest priority items on the team backlogs. However each involved executive influences the priority setting of a subset of product owners in the chain, resulting in priority mismatches over the front to back chain. Such conflicts results in impeded delivery of the front to back features.

"*In team A an end to end feature has highest priority, at team B however a dependent story has low priority due to interdependent stories that need to be delivered first"*, IT development manager

Interviewees state that priority conflicts are expected to be resolved since front to back priority setting has been predefined by a roadmap manager for 2014.

"*From now on we have a roadmap manager at strategic level which has defined the top 10 strategic programs which will be executed in 2014",* IT development manager

Even though at the strategic level clear priorities have been set, alignment issues between teams occur, such as difficulties to align the definition of done and testing activities:

"Yes we have a definition of done, but try to align that over the full chain with clear requirements and acceptance criteria", Product owner

The development process in each of the teams has a certain unpredictability which is leveraged by the chain setting. In case one team cannot deliver, the deployed feature is delayed:

"*Misalignment in timing between teams happens regularly. Recently I had a feature which was on the list of each involved team. However one of the teams that had delays earlier now experienced test defects, while having a due deadline of the code freeze. At the end of the sprint nothing was delivered"*, project manager

4 Conclusion

Applying Scrum in an interdependent application chain seems not an easy task. We found four main types of issues in chains of codependent Scrum teams: (1) issues in the alignment between teams, (2) mismatching backlog priorities, (3) lack of coordination in the chain and (4) unpredictability of committed deliveries.

The first type concerns alignment issues between teams, mainly related to a different way of working. A lot of these issues express themselves during end to end testing, yet a lot of issues find their origin earlier in the development lifecycle.

The second type concerns backlog priority conflicts at the strategic level which cascade to the operational level. As teams have an intra-team focus, each team develops the application in accordance with its backlog. However, given the application interdependencies, features are only ready to be delivered if all required teams deliver the necessary application changes.

The third type - coordination - indicates boundary spanning activities over the front to back chain. Such boundary spanning supports information and knowledge sharing between the involved teams.

The fourth type concerns unpredictability during the lifecycle process. Feature delivery, involving multiple teams, can only take place in case all interdependent application changes are delivered. If delivery in one of the teams is impeded (e.g. caused by misinterpretation or unexpected work) the feature is not delivered in that sprint, delaying time to market.

The results indicate that Scrum applied in application chains can result in increased delivery risk, despite the Scrum goal of decreasing delivery risk.

References

1. Agarwal, M., Majumdar, R.: Tracking Scrum projects Tools, Metrics and Myths About Agile 2(3) (2012)
2. Freudenberg, S., Sharp, H.: The top 10 burning research questions from practitioners. IEEE Software 27(5), 8–9 (2010)
3. Dingsøyr, T., Moe, N.B.: Research challenges in large-scale agile software development. ACM SIGSOFT Software Engineering Notes 38(5), 38–39 (2013)
4. Rising, L., Janoff, N.S.: The Scrum software development process for small teams. IEEE Software 17(4), 26–32 (2000)
5. Leffingwell, D.: Scaling software agility: best practices for large enterprises. Addison-Wesley Professional (2007)
6. Sutherland, J.: Future of scrum: Parallel pipelining of sprints in complex projects. In: Agile Conference, 2005. Proceedings. IEEE (2005)
7. Harland, C.: Supply chain management, purchasing and supply management, logistics, vertical integration, materials management and supply chain dynamics. Blackwell Encyclopedic dictionary of operations management, p. 15. Blackwell, UK (1996)
8. Lejeune, M.A., Yakova, N.: On characterizing the 4 C's in supply chain management. Journal of Operations Management 23(1), 81–100 (2005)
9. van der Vaart, T., van Donk, D.P.: A critical review of survey-based research in supply chain integration. International Journal of Production Economics 111(1), 42–55 (2008)
10. Cao, M., Zhang, Q.: Supply chain collaboration: Impact on collaborative advantage and firm performance. Journal of Operations Management 29(3), 163–180 (2011)
11. Wadhwa, S., et al.: Effects of information transparency and cooperation on supply chain performance: a simulation study. International Journal of Production Research 48(1), 145–166 (2010)
12. Saunders, M., Lewis, P., Thornhill, A.: Research methods for business students. Prentice Hall (2009)
13. Dul, J., Hak, T.: Case study methodology in business research. Routledge (2012)

Author Index