Georg Herzwurm
Tiziana Margaria (Eds.)

Software Business

From Physical Products to Software Services and Solutions

LNBIP 150

4th International Conference, ICSOB 2013
Potsdam, Germany, June 2013
Proceedings

Springer

Lecture Notes
in Business Information Processing 150

Series Editors

Georg Herzwurm
Tiziana Margaria (Eds.)

Software Business

From Physical Products to Software Services and Solutions

4th International Conference, ICSOB 2013
Potsdam, Germany, June 11-14, 2013
Proceedings

Springer

Volume Editors

Georg Herzwurm
University of Stuttgart
Stuttgart, Germany
E-mail: herzwurm@wi.uni-stuttgart.de

Tiziana Margaria
University of Potsdam
Potsdam, Germany
E-mail: margaria@cs.uni-potsdam.de

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-642-39335-8 e-ISBN 978-3-642-39336-5
DOI 10.1007/978-3-642-39336-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013941480

ACM Computing Classification (1998): K.1, K.6, D.2

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)

Message from the General Chair

Over the last five decades, software has become an integral part of society, allowing for productivity gains as well as providing means for serving customers and citizens in ways that would have been unimaginable to our grandparents, our parents, and even ourselves only a few years ago. Although it is easy to think of software as "just technology," in industry after industry it is becoming abundantly clear that software is the key area of innovation. For instance, for companies in the telecom industry, software R&D accounts for 80% of the R&D budget, whereas in the automotive industry, 70% of all innovations are driven by software.

Although many conferences focus on the technology aspects of software, few focus on the relationship between business and software and as well as the business of software. This is the purview of ICSOB. Over the last four instances of the conference, we have studied a wide variety of topics in this scope, but often concerned with the transformation of many businesses and industries toward selling software-enabled services and products where the software is the main value-providing element. The implications of this transformation are profound, including changes to the business model and strategy, system architecture, ways of working, tools and processes, as well as the way the company is organized.

For a conference to be successfully organized, however, it takes the proverbial village. As General Chair for ICSOB 2013, I am indebted to a great team. The Program Chairs, Tiziana Margaria and Georg Herzwurm, did a fabulous job in developing a very compelling program, helped by their Program Committee. Slinger Jansen acted as Workshop Chair and did a great job both soliciting highly interesting workshops and helping the workshops exploit the synergies between them. The Industry Chairs Asuman Sünbül and Barry D. Floyd worked as a great team and pulled together a very interesting and innovative industry day with strong appeal for the industry, ranging from start-ups to embedded systems companies. Local organizations were dealt with by Tiziana Margaria and her team and the website was managed by Eetu Luoma. Finances were managed by Petros Stratis through his Easyconference team. Finally, the Steering Committee, chaired by Sjaak Brinkkemper, provided the stability and organizational memory that also this conference had the pleasure of benefiting from.

We gratefully thank our sponsors: the European Patent Office (EPO) and the International Software Product Management Association (ISPMA), as well as the University of Potsdam, the gracious host for the conference.

I hope the proceedings of the conference offer readers valuable new insights that hopefully contribute to their research or software business.

June 2013 Jan Bosch

Message from the Program Chairs

Advancements in the software industry have had a substantial impact not only on productivity and on gross domestic product growth globally, but also on our daily work and life. Moreover there is a noticeable spillover within other industries (e.g., manufacturing) enabling new business models: Companies bundle their physical products and software services into solutions (e.g., using subscription models) and start to sell independent software products in addition to physical products. 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. Moreover, recent developments such as the emerging app economy offers a variety of opportunities for entrepreneurs or start-up companies.

To acknowledge the importance of these topics we have chosen the main theme "From Physical Products to Software Services and Solutions" for the 4th International Conference on Software Business (ICSOB 2013) held in Potsdam (Germany) on the premises of the Universität Potsdam, Campus Griebnitzsee, during June 11–14, 2013. ICSOB is a series of annual conferences born in 2010. The previous conferences were held in Boston (USA), Brussels (Belgium), and Jyväskilä (Finland).

For the main conference we received 44 research paper submissions from all over the world. Each paper went through a review process by at least three reviewers. The Program Committee deliberated with all the reviews and accepted 15 submissions to be presented as full papers at the conference (thus giving it an acceptance rate of 34%). In addition, eight papers were accepted as short research papers, seven of which are included in this book. The accepted papers follow diverse methodologies, and represent the diversity in research in our community. We have organized the papers according to the following categories:

- Cloud Computing
- Entrepreneurship and Start-Up Companies
- IT Markets and Software Industry
- IT Within Organizations
- Software Business Models and Business Process Modeling
- Software Platforms and Software Ecosystems
- Software Product Management

We are particularly indebted to the keynote speakers, Gregor Engels from University of Paderborn, who spoke about "On-the-Fly Computing - the Service-Oriented Software Market of the Future", and Almer Podbicanin from SAP, who

spoke about "From Customer-Specific Solutions to Products and Standardized Services".

In addition to the research paper presentations, we opened the conference on June 11 with:

- IWSECO, the 5th International Workshop on Software Ecosystems, organized by Carina Alves, Geir Hanssen, and Jan Bosch
- IW-LCSP, the workshop on From Start-Ups to SaaS Conglomerate: Life Cycles of Software Products, organized by Krzysztof Wnuk, Sami Hyrynsalmi, Maya Daneva, and Tuomas Mäkilä

We hosted an Industry Day on June 14 and a Doctoral Symposium on June 12, while the Industrial Exhibition spanned June 12 and 13.

As chairs of the Program Committee, we would like to thank the Program Committee members for their time and dedication in providing feedback to the authors. Their input helped shape this conference and maintain a high quality of research. As has been the case in previous conferences, the Steering Committee was an invaluable source of organizational memory and provided valuable guidance at critical junctures.

We also appreciate all the contributions given by the student volunteers, coordinated in Potsdam by our local Organizing Chair Henning Bordihn, and are indebted to Anna-Lena Lamprecht and Tobias Tauterat, who were essential in getting the proceedings ready in time: only thanks to all these individual and collective efforts could we ensure the success of this conference.

June 2013

Georg Herzwurm
Tiziana Margaria

Organization

Committees

General Chair

Jan Bosch	Chalmers University of Technology, Sweden

Program Chairs

Georg Herzwurm	University of Stuttgart, Germany
Tiziana Margaria	University of Potsdam, Germany

Industry Chairs

Barry D. Floyd	University of Potsdam, Germany
Asuman Sünbül	SAP AG, USA

Workshop Chair

Slinger Jansen	Utrecht University, The Netherlands

Doctoral Symposium Chairs

Anna-Lena Lamprecht	University of Potsdam, Germany
Tobias Tauterat	University of Stuttgart, Germany

Finance Chair

Petros Stratis	EasyConferences Ltd, Cyprus

Local Organizing Chair

Henning Bordihn	University of Potsdam, Germany

Web Chair

Eetu Luoma	University of Jyväskylä, Finland

Steering Committee

Sjaak Brinkkemper	Utrecht University, The Netherlands
Michael Cusumano	Massachusetts Institute of Technology, USA
Bala Iyer	Babson College, USA
Slinger Jansen	Utrecht University, The Netherlands
Pasi Tyrväinen	University of Jyväskylä, Finland
Björn Regnell	Lund University, The Netherlands
Inge van de Weerd	Utrecht University, The Netherlands

Program Committee

Sjaak Brinkkemper	Utrecht University, The Netherlands
Peter Buxmann	Darmstadt University of Technology, Germany
Michael Cusumano	Massachusetts Institute of Technology, USA
Barry D. Floyd	University of Potsdam, Germany
Samuel Fricker	Blekinge Institute of Technology, Sweden
Georg Herzwurm	University of Stuttgart, Germany
Thomas Hess	LMU Munich, Germany
Bala Iyer	Babson College, USA
Slinger Jansen	Utrecht University, The Netherlands
Thomas Kude	University of Mannheim, Germany
Patricia Lago	VU University Amsterdam, The Netherlands
Ulrike Lechner	Universität der Bundeswehr München, Germany
Olaf Mackert	SAG AG, Germany
Tiziana Margaria	University of Potsdam, Germany
Waldemar Meinzer	Volkswagen AG, Germany
Wolfram Pietsch	FH Aachen, Germany
Karl Michael Popp	SAP AG, Germany
Björn Regnell	Lund University, Sweden
Dirk Riehle	Friedrich-Alexander-University, Germany
Ina Schieferdecker	Freie Universität Berlin, Germany
Katarina Stanoevska-Slabeva	University of St.Gallen, Switzerland
Pasi Tyrväinen	University of Jyväskylä, Finland
Yoshimichi Watanabe	University of Yamanashi, Japan

Table of Contents

Software Business Models and Business Process Modeling

The Impact of Software Business Model Characteristics on Firm Performance 1
Markus Schief, Anton Pussep, and Peter Buxmann

Second-Order Servification 13
Johannes Neubauer and Bernhard Steffen

Sustainable Business Models for Services Using Semantic Web Components: Insights from the Field 26
Mary Tate and Elfi Furtmueller

IT Markets and Software Industry

Measuring Sales Cannibalization in Information Technology Markets: Conceptual Foundations and Research Issues 31
Francesco Novelli

Determinants and Dynamics of Technology-Related Acquisitions: The Case of Software-Based Industries 43
Marcus Wagner

IT within Organizations

Engineering Open Innovation – Towards a Framework for Fostering Open Innovation 48
Krzysztof Wnuk and Per Runeson

Improving Businesses Success by Managing Interactions among Agile Teams in Large Organizations 60
Antonio Martini, Lars Pareto, and Jan Bosch

Current Trends in Employee Recruitment Using the Internet 73
Elfi Furtmueller

Software Product Management

Post-deployment Data Collection in Software-Intensive Embedded Products 79
Helena Holmström Olsson and Jan Bosch

A Model of Commercial Open Source Software Product Features 90
Florian Weikert and Dirk Riehle

A Framework for Strategic Positioning of IT-Products 102
Wolfram Pietsch

Cloud Computing

Cloud Services Pricing Models 117
Gabriella Laatikainen, Arto Ojala, and Oleksiy Mazhelis

The Impact of Software-as-a-Service on Software Ecosystems 130
Sebastian Walter Schütz, Thomas Kude, and Karl Michael Popp

Towards a Conceptual Framework for Assessing the Benefits of Cloud Computing 141
Nattakarn Phaphoom, Xiaofeng Wang, and Pekka Abrahamsson

Entrepreneurship and Startup Companies

The Importance of the Business Idea for New Venture Creation in the Software Industry 153
Natalie Kaltenecker, Christian Hoerndlein, and Thomas Hess

Exploring How Feature Usage Relates to Customer Perceived Value: A Case Study in a Startup Company 166
Sarunas Marciuska, Cigdem Gencel, and Pekka Abrahamsson

Business Incubation Practices and Software Start-up Success in Turkey 178
Gozem Guceri-Ucar and Stefan Koch

Software Platforms and Software Ecosystems

Ecosystem Health of Cloud PaaS Providers 183
Garm Lucassen, Kevin van Rooij, and Slinger Jansen

Defining App Stores: The Role of Curated Marketplaces in Software Ecosystems 195
Slinger Jansen and Ewoud Bloemendal

Towards Platform-Based Enterprise Systems – Conceptualization and Research Directions 207
Carl Simon Heckmann and Alexander Maedche

Software Ecosystem Roles Classification 212
Eko Handoyo, Slinger Jansen, and Sjaak Brinkkemper

Formal Description for SaaS Undo 217
Hernán Merlino, Oscar Dieste, Patricia Pesado, and Ramón García-Martínez

Doctoral Symposium

Virtual Character Based Interactive Interfaces for Deaf and Functionally Illiterate Users 223
Nadeem Ahmad

Simplicity in Application Development for Business Model Design 225
Steve Boßelmann

Software Ecosystem Modeling 227
Eko Handoyo

Impact of Enterprise System Modularity on Process Performance 229
Carl Simon Heckmann

Managing Speed in Companies Developing Large-Scale Embedded Systems 231
Antonio Martini

Global Manufacturing Networks as Software-Intensive Service Providers Motivation, Relevance, Research Objective 233
Tobias Tauterat

Author Index 235

The Impact of Software Business Model Characteristics on Firm Performance

Markus Schief, Anton Pussep, and Peter Buxmann

SAP Research, Dietmar-Hopp Allee 16, 69190 Walldorf, Germany
markus.schief@sap.com
Technische Universität Darmstadt, Chair of Information Systems, Hochschulstraße 1, 62849 Darmstadt, Germany
{Pussep,buxmann}@is.tu-darmstadt.de

Abstract. Business models have become a topic of increasing academic interest and have emerged as a unit of analysis for performance studies. The software industry has been the source of major business model innovations and is hence of particular interest to researchers and practitioners. In this paper we collect business model data for 120 public U.S. software firms. While some data can be retrieved from Thomson Reuters database, most variables specific to the software firms are obtained from a tedious expert classification of 10-K and 20-F annual reports. The results show that the business model variables under study significantly impact financial performance, but are hardly reflected in market performance. Thus, they determine firm success, but do not necessarily affect investor decisions. Our cross-disciplinary research is rooted in the fields of strategic management and software business. We contribute by providing insights into business model characteristics and the determinants of software firm performance.

Keywords: business model, performance, firm, software industry.

1 Introduction

The question what drives firm performance has received considerable research attention and remains of continuous interest in strategic management [1] and other disciplines such as information systems [2]. In the beginnings of the field, researchers examined the impact of economic and industry factors on performance [3]. Today, it is generally agreed that this traditional focus on industry factors ignores the fact that firms can make discretionary choices [4]. Short et al. [1] conducted a simultaneous analysis of the industry, group, and firm level and found that, when examined together, the impact of the firm level is the strongest.

In order to identify the determinants of firm performance, often a broad set of firm characteristics is analyzed. However, previous studies have used very different sets of these characteristics. Capon et al. [5] provided an overview and found that while some characteristics have been studied extensively others have been neglected. Recently, business models have emerged as a unit of analysis and became a topic of increasing

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 1–12, 2013.

academic interest [6]. There is increasing consensus that business models can also impact firm performance [7]. Their characteristics thus represent additional potential determinants of firm performance.

While business models provide a promising area for firm performance research, the constituents of the business model concept itself are still a question of academic debate. Business model performance studies often use an industry-specific setting [7] offering the opportunity to provide specific business model concepts and operationalizations. Strategic management research also acknowledges that an industry-specific setting is required for meaningful conclusions [8]. One reason is that industry-specific variables are more meaningful in delineating firms. Another reason is that researchers require deep knowledge of the particular industry under study to derive conclusions.

The software sector is of particular interest to researchers and practitioners. Software products and markets are characterized by a number of specifics (e.g., ease of replication and network effects), these specifics provide a unique setting for research [9]. For practitioners, the determinants of firm performance are of particular interest as the dynamics of the software industry open up great opportunities for firm growth and profitability. The software sector hence provides an interesting setting for an analysis of the relationship between business models and firm performance.

According to the literature review by Lambert and Davidson [7] most empirical studies focus on business models of Internet firms (e.g. [2] [10]). To date, only few empirical studies have analyzed the performance of software firms. Three studies investigate the business model impact on software firms' performance in particular. Engelhardt differentiates [11] four business model classes (general software and services, business software, specialized software, and internet software) and reports (based on a German firm sample) significant performance differences in terms of sales growth and productivity growth. Valtakoski and Rönkkö [12] examine the impact of eight business model classes (software product, deployment project, development service, ASP and SaaS, content and ads, software consulting, hardware, and non-software firms). For the Finnish software industry they report significant performance differences in terms of revenue growth, profitability, and productivity. Rajala and Westerlund [13] analyze the impact of two business model characteristics (customer proximity and product uniformity) on financial and market performance. Again, they report significant performance effects for Finnish software firms.

While business models have been found to impact software firm performance, additional studies are needed that further specify the business model concept (beyond fixed number of business model classes or low number of characteristics) and broaden the geographical coverage (beyond Germany and Finland). Our research builds upon this previous work and links the fields of software business and strategic management by asking the following research question:

Which software business model characteristics determine software firm performance?

As firm performance is a multi-dimensional concept, it is widely accepted that multiple measures must be taken into account. In general, two sets of characteristics, financial (accounting-based) and market (capital market-based) performance

measures can be differentiated [1], [10]. While financial measures represent values of realized performance documented in accounting books, market measures indicate the perceived performance by investors. Market measures are sometimes also viewed as long-term measures [10]. We thus derive and test the following hypotheses:

H1: Financial performance varies systematically with differences in software business model characteristics.

H2: Market performance varies systematically with differences in software business model characteristics.

In order to test the hypotheses and answer our research question, we analyze five software industry-specific business model characteristics with detailed business model data of 120 software firms from annual reports. We estimate the impact of business model characteristics through regression models with different performance measures and control for generic strategic variables. Our hope is that this paper will contribute to the theoretical development of the business model concept and to the identification of determinants on firm performance in the software industry. Decision-makers will find the results useful as a source for strategy considerations.

The remainder of this paper is organized as follows. Next section describes the selection of the sample firms, the applied method as well as the operationalization of business model characteristics and performance measures. Then, the results with regard to our previously defined hypotheses are presented. We proceed with the discussion of our findings and answer our research question. Finally, we conclude the paper and provide avenues for further research.

2 Method

2.1 Sample

The sample firms are drawn from the Bureau van Dijk Orbis database. All financial data is retrieved from the Thomson Financial Worldscope database.

We select active, publicly listed firms by their primary Standard Industry Classification (SIC) code, within the range 7371-7374. According to code descriptions, the codes identify software firms and have been applied in other software industry performance studies [14]. The sample is further limited to firms with no single shareholder having a direct stake over 50 percent. These firms are less exposed to external control, such that the firm's performance can be better attributed to its strategic choices. As we rely on expert classification of 10-K and 20-F annual reports (these are required by the Securities and Exchange Commission, SEC) as a data source for software business model variables, the sample is restricted to firms listed at a U.S. stock exchange. Finally, we are forced to limit the number of firms, as the expert classification of reports is very time consuming. For that, we select the 120 firms with the highest revenues. In 2010, the firms in our sample yielded annual revenues between $US165 million (Sourcefire) to $US107bn (IBM).

2.2 Software Business Model Variables

In our study we derive software industry-specific business model variables from the concept proposed by [15]. They define a comprehensive business model framework for software firms based on a broad set of economic properties attributed to the software industry. We are forced to restrict the number of variables as not all business model characteristics qualify for an analysis based on secondary data. Finally, we define five variables. Thereof, four variables can be retrieved via expert classification of annual reports and one variable from a financial database. The detailed operationalization of each variable is described in Figure 1.

Variable	Options	Decision basis	Decision rule	Decision by	Source	SEC Items	Key Words	Comment
Product vs. Services	Product	Segment revenues	>=50%	Expert	Annual Reports	Item 1, 6, 7, notes	n/a	Percentage of a firm's revenue streams stemming from products (software licenses or physically tangible products such as hardware) versus services (maintenance, support, consulting, training, software as a service, and other services).
	Service		>50%					
Product Focus	Infrastructure	Segment revenues	>=50%	Expert	Annual Reports	Item 1, 6, 7, notes	n/a	Focus of a firm's products and services in terms of a software stack layer. We differentiate between infrastructure and application software. For that we refer to the taxonomy by Forward and Lethbridge (2008); The following domains belong to Infrastructure: A.des 1-4; B; and C. Others are classified as Applications.
	Application		>50%					
Target Customer	B2C	Segment revenues	>=50%	Expert	Annual Reports	Item 1, 7, notes	customer, consumer	Main customer group of the delivered products/services. While B2C customers use the products/services for their private purposes, business customers (B2B) use it for pursuing their business.
	B2B		>50%					
Target Industries	Focus	Number of adressed industries	One or few	Expert	Annual Reports	Item 1	industry, industries	Number of adressed industries, While focused companies only address one or a few industries, diversified firms address various industries.
	Broad		Many					
Payment Flow Structure	Metric	n/a	n/a	n/a	Thomson Reuters	n/a	n/a	Deferred short- and longterm revenues divided by total sales.

Fig. 1. Assignment rules for software business model variables

The derived variables are specific to the software industry and may not qualify for other industries without adjustments. These variables require intimate knowledge of the software industry and are special in at least one of the following characteristics: (1) The variable deals with an aspect that may not be relevant to other industries (e.g. target industries). (2) The definition and terminology of options are highly industry-specific (e.g. product focus on application or infrastructure software). (3) The assignment rules need to be specifically formulated for the industry (e.g. for product vs. service; all potential software and associated products/services need to be considered and documented in the assignment rule). (4) Expert knowledge is required to conduct the classification as the nature of software needs to be explored (e.g. analyze from which target customers a product can be used and examine the distribution of associated revenues).

With respect to the first four variables, we build on the common technique in strategic management research of using expert panelists [17]. Three experts independently code 10-K or 20-F annual reports of the selected 120 firms. Our

strategy is that each annual report is independently classified by two experts. Consequently, each coder does 320 classifications (4 variables times 80 firms). Then, we consolidate all ratings. Inter-rater reliability is analyzed by conducting a pair-wise comparison of responses for all firms. In sum, there are 480 opportunities for disagreement in recording the variables configurations. Disagreement occurs 109 times, so the rate of overall initial coding agreement is 77.29 percent. Beyond examining overall agreement, we analyze the agreement between each pair of experts by calculating Cohen's Kappa [18]. Initial coding agreement between expert one and two is 60.18 percent, 49.04 percent for expert one and three, and 58.77 percent for expert two and three. Coding agreement is finally reached through discussion of the non-consistent 109 cases until mutual agreement is reached. All initial differences are resolved through discussions, so the final agreement is 100 percent. However, the final sample only consists of 454 ratings. For 26 items, no classification is possible.

One further business model variable, payment flow structure, is drawn from the Thomson Reuters Worldscope database. We define payment flow structure as a firm's deferred revenues divided by sales. This ratio reflects the degree of revenues that cannot be immediately booked as revenues in the profit and loss statement but needs to be deferred to the balance sheet. The proportion of revenue that is deferred may mainly depend on two aspects in the software industry [19]. First, in a multiple-element software arrangement, firms may only book sales if vendor specific objective evidence (VSOE) of fair value for each bundle element exists. Secondly, in case of recurring subscription models, payments may not be recorded as sales before the ongoing delivery has occurred. Then, parts of the payment need to be deferred.

2.3 Control Variables

To incorporate results of previous firm performance studies, we include control variables in our study setting. Previous literature has used an unmanageable amount of variables [5]. In general, variables can be organized in two strategic dimensions, impacting firm performance: business scope and resource commitment [1]. Business scope is determined by the markets in which a firm operates [8]. In turn, resource commitment is determined by unique resources and capabilities of a firm [20].

While our applied business model characteristics cover the business scope of a firm, our control variables cover the resource commitment of a firm. In contrast to the business model variables, the control variables have been derived from previous studies and are generic in the sense that they are not specific to the software industry, but can be found across all industries. We acknowledge that the selection of control variables is not exhaustive because there are simply too many potential control variables. For instance, we do not include R&D expenditures because this balance sheet position is defined very differently across software firms. Our selection is limited to three variables:

- *Size* is measured as the natural log of sales [1]. Size is considered as an indicator for network effects and opportunities for economies of scale [21]. It is probably the most widely included variable in firm performance studies.

- *Capital intensity* is calculated as capital expenditures divided by sales [1]. A firm may commit capital expenditures in order to build up property and equipment [1]. Capital intensity is likely to correlate with innovation [21].
- *Vertical integration* is measured as the difference between sales and cost of goods sold, divided by sales [22]. A firm may decide to focus on few activities or provide a one-stop shop to its customers by covering the entire value chain. A high degree of vertical integration indicates high control over customers and suppliers [21].

2.4 Performance Variables

We use multiple measures to capture firm performance which are commonly used in this field of research [1, 10]. We capture financial performance with operating profit margin (OPM) and return on assets (ROA). Market performance is captured with Tobin's q (calculated as the sum of market value and debt, divided by total assets). All data is obtained from the Thomson Reuters database for the year-ends of 2009, 2010, and 2011. We average the performance data for these three years in order to smooth out short-term trends [8]. Using the three-year average further accounts for a certain time lag between strategic decisions and performance effects.

2.5 Statistical Analysis Applied

The data in the present study is analyzed and hypotheses examined through OLS regression analysis, which is the dominant method to investigate the determinants of financial performance [5]. To improve the validity of our model, we perform a logarithmic transformation of one variable (size) and test for multicollinearity problems among independent variables. We find that variables correlate reasonably low (maximal value being 0.53 and all other values below 0.3) and hence qualify for further analyses. Finally, we run the OLS multiple regression procedure as method for the estimation of proportions explained by each independent variable in the variation of the dependent ones. To calculate parameter and fit estimates, we use R software.

3 Results

3.1 Descriptive Results

Our sample firms have an average age of 25 years. 76 percent of the firms generate a major share of their revenues with services rather than products. Further, 61 percent of them focus on application and 39 percent on infrastructure software. 94 percent of the firms mainly sell to business customers. In addition, 65 percent predominantly serve a broad set of industries instead of focusing on few target segments. Finally, the average ratio of deferred revenues to sales is 21 percent, indicating that the majority of revenues can be recorded at the point of payment instead of deferring them.

3.2 Regression Analyses

The results of our multiple regression analyses are presented in Table 1. We calculate one model for each of the three performance variables. With respect to our hypotheses, we derive the following conclusions. We find support for H1 (significant impact of software business model characteristics on financial performance) in Model 1 and 2. Overall, this set of variables explains 40 percent of the variance in OPM and 15 percent in ROA. For business model variables, we find evidence that those firms offering infrastructure software, addressing B2B customers, selling to few specific industries and recognizing revenues upfront yield significant positive results. With respect to the control variables, capital intensity has a negative impact, while size and vertical integration have a positive impact on financial performance.

Table 1. Multiple regression analyses results

	Financial Performance		**Market Performance**
	ROA	**OPM**	**Tobin's Q**
	Model 1	Model 2	Model 3
Constant	-4.55	-24.99	2.19 *
Control Variables			
C1: Capital Intensity	-0.52 *	-0.81 **	-0.03
C2: Size	0.44	2.98 ***	-0.31 **
C3: Vertical Integration	10.16 *	23.95 ***	3.64 ***
Business Model Variables			
BM1: Product vs. Service	2.89	2.34	0.87 *
BM2: Product Focus	-2.37	-5.32 *	-0.49
BM3: Target Customer	7.83 *	14.24 **	0.43
BM4: Target Industries	-3.42 *	-5.98 *	-0.54
BM5: Payment Flow Structure	-8.66 *	-10.14 †	-1.21
R^2	0.23	0.46	0.29
Adj. R^2	0.15	0.40	0.22
df	83	83	78
F	3.03 **	8.70 ***	3.95 ***

**** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$; † $p < 0.1$*

The second dimension deals with market performance. With respect to H2 (significant impact of business model characteristics on market performance), we find hardly any support in Model 3. Only one business model variable (product vs. service) is significant. In contrast, market performance varies systematically with differences in our control variables. Overall, the variables explain 22 percent of the variance in Tobin's q. We find evidence that small firms offering services and following a high vertical integration strategy yield significant positive results.

4 Discussion

4.1 Insights from Financial Performance

Overall, the model quality and significance levels for OPM are higher than for ROA. It seems that the standardization through asset division decreases statistical power. A rationale for this result may be the nature of a digital goods industry, which is less asset-centric. While variances in asset bases are hence less distinctive, the usage of ROA does not reveal additional insights compared to OPM.

Looking at the control variables, vertical integration shows a very significant impact. The role of vertical integration as a factor explaining differences in profitability has been widely tested in other studies [5]. It hence seems to be lucrative to cover as many software value chain activities as possible and to offer a holistic one-stop shop solution portfolio. This may also be one rationale for the high rate of mergers and acquisitions in the software industry [23]. Besides, bigger firms yield higher operating profit margins. This finding is in line with the expected networked effects in the software industry [9]. Further, we find capital intensity as a negative performance driver. The role of capital intensity as a factor explaining differences in profitability has been widely tested as well [5]. Those firms being capital intensive in a digital goods industry underperform their peers. In sum, all control variables impact financial performance significantly.

With respect to the software business model variables, our results reveal that infrastructure companies achieve better performance than application software providers. A reason for that might be that application software usually is more customer-specific than infrastructure software. The latter may hence attract a broader customer base and achieve higher economies of scale. This goes in line with a statement by Gao and Iyer [14] claiming that the main underlying factor for success is the ability to establish platforms with high levels of integration and high associated switching costs for users. A further significant variable is target customer. Firms mainly selling to end-users, instead of businesses, achieve significant worse OPM and ROA. A rationale for this finding might be that consumers have a lower willingness to pay as they are not as dependent on software as businesses. Moreover, consumers might be more willing to use free of charge open source offerings or illegal copies (e.g. games). Another interesting result is the impact of the payment flow structure. Firms with a high rate of deferred revenues (e.g. referring to recurring payments) perform worse than their peers being able to charge initial upfront license fees. From an annual statement perspective, firms with initial license revenues can record them as realized revenues in the profit and loss statement (instead of putting deferred revenues to the balance sheet) and hence increase their OPM and ROA. Finally, the focus on few dedicated target industries yields positive results. Industry-specific offerings seem to offer more value for customers and result in higher software vendor margins.

All in all, we can conclude that the business model variables under study are highly relevant for software firms' financial performance. While OPM varies systematically with differences in four of five business model variables, ROA varies in three. These variables can thus be considered to impact software firms' financial performance.

4.2 Insights from Market Performance

Looking at the control variables, size and vertical integration are the dominant drivers of market performance measured as Tobin's q that represents a derivate of a firm's price-to-book ratio. Notably, capital markets seem to appreciate smaller software companies. This finding may refer to the fact that the software industry is a highly dynamic sector with high innovation and short product lifecycle rates [9]. So, investors appreciate firms challenging the incumbent players in the software sector. Besides, vertical integration achieves significant positive results. Highly integrated companies might hence be seen by investors as well-prepared to compete in the industry by positioning themselves as one-stop-shops offering end-to-end solutions.

With respect to the business model variables, we find a positive impact of companies predominantly focusing on software services instead of software products. Since service revenues in our study comprise maintenance as well as software-as-a-service (SaaS) revenues, investors may appreciate firms, which are less dependent on cyclical product sales or firms which move towards a SaaS operating model.

Apart from this variable, the business model variables under study do not show any further significant impact on market performance. Thus, while the examined business model characteristics seem to impact financial performance, they are hardly reflected in market performance. In spite of their impact on financial performance, business model characteristics may hence not necessarily anticipated to the same extent by market investors. We believe that the availability of information may explain this finding. As business model variables are more difficult to gather in a standardized format for many enterprises, the markets are less aware of them. Investors predominantly refer to public news and data that is available in financial databases.

4.3 Implications for Researchers

This paper contributes to the long tradition of performance research [24] and provides linkages between strategic management and software business. These fields are interlinked through the usage of domain-specific variables in the context of strategy and firm performance. It appears to us that software business research can benefit from the findings that are known in strategy research, while strategy research could benefit from a deeper understanding of characteristics that are highly domain-specific in nature.

Our results suggest that software industry-specific business model variables need to be taken into account. Interestingly, the degree and statistical power of the impact depends on the performance measures under study. Whereas the impact on OPM is strong, the impact on ROA and Tobin's q is less significant. In a digital goods industry context, the standardization through division of total assets or a firm's book value, respectively, seems to decrease the quality of the models. Moreover, capital markets do not necessarily reflect the impact of business model characteristics.

The performance determinants can be compared to other single-industry studies. The comparison might be difficult when comparing industries based on different (since domain-specific) variables. However, further studies of digital goods industries

should follow, just as banks have turned out to be of continuous interest in the field of strategy. The banking sector has been widely analyzed as detailed data is available. The validity of a generalization to other sectors remains questionable. Our results show that for some business model characteristics detailed data can be gathered from software firms' annual reports. This data supports analysis of the industry in the context of performance drivers.

4.4 Implications for Practitioners

This research makes contributions that are relevant to decision-makers in software firms as well as to investors interested in software firms. Our study offers at least two useful insights for software firm managers being understandably curious about which strategic configuration is most profitable. First of all, this study emphasizes the importance to take business model variables into consideration, particularly with respect to the financial performance of software firms. So, a firm's strategic positioning in an industry has a significant impact on operating margins and on return on assets. It is important for managers to reflect the own firm characteristics and to compare them with competitors and partners. Secondly, financial markets hardly seem to anticipate business model characteristics. Consequently, managers need to foster the communication of strategic advantages. We hope that managers can use this data to understand at a deeper level the structural choices they have and how to manage them effectively. Nonetheless, our study can only provide a foundation for the normative question of how individual firms can exploit or modify their strategies to improve their performance.

We further think that determinants of market performance yield important results to investors in software firms. Our results suggest that investors should anticipate business model characteristics beyond generic performance drivers. While the latter can be retrieved from financial databases, structured and standardized business model classifications may not be as easily accessible for the software industry. Usually, business model characteristics are retrieved from news sources in a low standardized format. Our firm classification can provide a valuable structured software industry overview that may lead to novel insights about investment opportunities.

5 Conclusion

The purpose of this study is to explore the impact of business model characteristics on software firms' performance. This paper interlinks the fields of software business and strategic management and contributes two main findings. Firstly, we provide insights on determinants of firm performance in the software industry. Secondly, we demonstrate that business model variables provide additional explanation of variance in financial performance. This set comprises variables on the structure of what firms actually do by including domain-specific characteristics.

We find support for significant effects of business model variables on financial performance. While this holds true for OPM and ROA, the impact on OPM is

stronger than on ROA. Standardization through the amount of assets seems to be inappropriate for a digital goods industry such as the software sector. In contrast to the impact on financial performance, our results do not support the hypothesis that business model characteristics are reflected by market performance. Business model information is not as easily available as data on other common performance drivers. Nevertheless, reflecting the impact on financial performance, ultimately, business model characteristics should also impact a firm's market performance.

Decision-makers in software firms will find our results useful for competitor analysis. The performance analysis allows them for tweaking their strategies to the high-performance characteristics. Moreover, investors are provided with insights for investment analysis. A structured industry overview in line with performance analyses can provide valuable insights for investment decisions.

There are several limitations to this study. As we rely on SEC reports, only public firms listed in the U.S. are included in our sample. Consequently, results may be impacted by the U.S. software market and conclusions with regard to other geographic areas may not be possible. Our sample is further limited to the 120 largest firms due to the complexity of data collection, which may result in a sample bias. The results are further limited by the accuracy of variables for the same reason. Four business models variables are reduced to binary scales in order to deal with the complexity and information scarcity. Further, the number of business model variables that can be retrieved from annual reports is limited. Likewise, large firms may have more than one business model and our classification only captures the dominant one. Our further research will focus on extending the sample, adding primary data sources, and analyzing data longitudinally.

Acknowledgments. This work was supported by the Software-Cluster and partially funded within the Leading-Edge Cluster competition by the German Federal Ministry of Education and Research (BMBF) under grant "01IC10S05".

References

1. Short, J.C., Ketchen, D.J., Palmer, T.B., Hult, D.T.M.: Firm, Strategic Group, and Industry Influences on Performance. Strategic Management Journal 28, 147–167 (2007)
2. Grover, V., Saeed, K.A.: Strategic orientation and performance of internet-based businesses. Information Systems Journal 14, 23–42 (2004)
3. Porter, M.E.:: Competitive strategy: Techniques for analyzing industry and competitors. Free Press, New York (1980)
4. Nelson, R.R.: Why Do Firms Differ, and How Does It Matter? Strategic Management Journal 12, 61–74 (1991)
5. Capon, N., Farley, J.U., Hoenig, S.: Determinants of Financial Performance: A Meta-Analysis. Management Science 36, 1143–1159 (1990)
6. Burkhart, T., Werth, D., Krumeich, J., Loos, P.: Analyzing the Business Model Concept — A Comprehensive Classification. In: 32nd International Conference on Information Systems, pp. 1–19 (2011)
7. Lambert, S.C., Davidson, R.A.: Applications of the Business Model in Studies of Enterprise Success, Innovation and Classification. European Management Journal (2012)

8. Mehra, A.: Resource and Market Based Determinants of Performance in the US Banking Industry. Strategic Management Journal 17, 307–322 (1996)
9. Buxmann, P., Diefenbach, H., Hess, T.: The Software Industry: Economic Principles, Strategies, Perspectives. Springer, Heidelberg (2012)
10. Zott, C., Amit, R.: The Fit Between Productmarket Strategy and Business Model: Implications for Firm Performance. Strategic Management Journal 26, 1–26 (2008)
11. Engelhardt, L.: Entrepreneurial Models and the Software Sector. Competition and Change 8, 391–410 (2004)
12. 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)
13. Rajala, R., Westerlund, M.: The Effects of Service Orientation, Technology Orientation and Open Innovation on the Performance of Software-intensive Service Businesses. In: 45th Hawaii International Conference on System Sciences, pp. 1532–1541. IEEE (2012)
14. Gao, L., Iyer, B.: Analyzing Complementarities Using Software Stacks for Software Industry Acquisitions. Journal of Management Information Systems 23, 119–147 (2006)
15. Schief, M., Buxmann, P.: Business Models in the Software Industry. In: Proceedings of the 45th Hawaii International Conference on System Sciences, pp. 3328–3337. IEEE, Maui (2012)
16. Forward, A., Lethbridge, T.C., Edward, K.: A Taxonomy of Software Types to Facilitate Search and Evidence-Based Software Engineering. In: Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research, pp. 1–13. ACM (2008)
17. MacCormack, A., Verganti, R., Iansiti, M.: Developing products on “Internet time”. Management Science 47, 133–150 (2001)
18. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement 20, 37–46 (1960)
19. Dechow, P., Skinner, D.: Earnings management: Reconciling the views of accounting academics, practitioners, and regulators. Social Science Research Network (2000)
20. Barney, J.: Firm Resources and Sustained Competitive Advantage. Journal of Management 17, 99–120 (1991)
21. Houthoofd, N.: Strategic Groups as Subsets of Strategic Scope Groups in the Belgian Brewing Industry. Strategic Management Journal 18, 653–666 (1997)
22. Hutzschenreuter, T., Gröne, F.: Changing Vertical Integration Strategies under Pressure from Foreign Competition: The Case of US and German Multinationals. Journal of Management Studies 46, 269–307 (2009)
23. Mergerstats free reports, http://www.mergerstat.com/newsite/free_report.asp
24. Schmalensee, R.: Do Markets Differ Much? The American Economic Review 75, 341–351 (1985)

Second-Order Servification

Johannes Neubauer and Bernhard Steffen

Chair of Programming Systems, Technische Universität Dortmund, Germany
{johannes.neubauer,steffen}@cs.tu-dortmund.de

Abstract. In this paper we present second-order servification, a business process modeling paradigm for variability. Key to this paradigm is to consider services and even whole subprocesses as 'resources' of a (second-order) business process, which can be created, selected, and moved around just like data. This does not only allow us to easily define new variants of a business process simply via second-order parameterization, but also to exchange its constituent services (and even sub-processes) dynamically at runtime. In fact, the concrete implementation of a second-order activity in a process model may be unknown when the process starts, and built-up and exchanged while the process is running. We will illustrate the ease of the new paradigm along a flight booking scenario, where our corresponding second-order process model allows us to dynamically instantiate the payment process even with process implementations that were not available when the overall process started.

Keywords: service orientation, servification, business process modeling, executable models, variability.

1 Introduction

In its beginning *(business) process modeling* (*BPM*), typically done with Event-Driven Process Chains [1] and supported by tools like ARIS [2], was merely a matter of business/requirement analysis and documentation. The often enormous documents, which, were helpful for clarificaton in early project phases, were not (directly) linked to any development artifacts during the realization of processes. So, they left what is commonly known as the *semantic gap* in software engineering. The service oriented development paradigm brought the promise of change: (business) process models should be refined until a realization in terms of intuitive services is possible [3], and indeed, the new version of the *Business Process Modeling and Notation* (*BPMN 2*) claims executability. One of the main remaining issues is the actual integration of arbitrary functionality into a service oriented environment (in particular business processes), called servification, whose state of the art is systematically discussed in [4]. This study reveals that only the so-called *domain-specific business activities*[1] support reusability and a sufficient abstraction from technological detail in a way that allows for

[1] We use the the notion "activity" for executable "nodes" or "vertices" as defined in the specification of BPMN.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 13–25, 2013.

agile, user-centric process development [5]. This sets them apart from *script-*, and *technical activities*. The former embed code directly into a process model, violating the concepts of *reusability* and *separation of concerns*, whereas the latter are strongly involved in technologies like *WS-**[2] [6,7,8] or *representational state transfer* (*REST*) [9]. Hence, script activities and technical activities pose an entry hurdle for non-programmers, who have e.g. to talk about a "web service endpoint implementing the service `BookAFlight`" instead simply of the business activity "book a flight".

In this paper we present second-order servification, a business process modeling paradigm for variability. This paradigm does not only allow the integration of arbitrary functionalities into our process modeling environment, but also to delay the decision about their concrete implementation until runtime. Key to this paradigm is to consider services and even whole subprocesses as 'resources' of a (second-order) business process, which can be created, selected, and moved around just like data. This does not only allow us

- to easily define new variants of a business process simply via second-order parameterization, i.e. via interface-conform exchange of functionality at design time, but also
- to exchange the processes constituent services (and even sub-processes) dynamically at runtime.

In fact, the concrete implementation of a second-order activity in a process model may be unknown when the process starts, and built-up and exchanged while the process is running without touching the processes code or model. Being based on formal interface specifications, our approach additionally guarantees the executability of all variants or runtime instantiations fully automatically, and, in particular, without any effort from the business process modelers' side. The required interface conformance is only based on an according servification process for each of the constituent services, which, in general, needs technical expertise. However, for semantically annotated service libraries, the required servification process has the potential to become largely automated.

The paper will end with an illustration of the ease of the new paradigm along a flight booking scenario, where our corresponding second-order process model allows us to dynamically instantiate the payment process even with process implementations that were not available when the overall process started.

After sketching the state of the art of BPM solutions in Sec. 2, we will present our second-order servification approach in Sec. 3, before Sec. 4 discusses a simple flight booking service scenario in a first-order setting. This service will then be used in Sec. 5 for illustrating the power of second-order servification by showing how situation-specific payment methods are put in place by exchanging process instances at runtime. Finally, Sec. 6 summarizes our conclusions and provides direction to future work.

[2] The notion WS-* describes a family of specifications related to web services.

2 BPM: State of the Art

First attempts to standardize BPM like the *business process execution language* [10] (*BPEL*) – in combination with *business process modeling notation*[3] (*BPMN*) [11,12,13] adding a graphical representation [14,15,16,17] – are language independent, but bind to a technology like WS-*. As a consequence, any functionality that should be accessible through the environment has to be wrapped into a service of the corresponding technology [18]. To tackle this issue the *Object Management Group* (*OMG*) published the technology agnostic standard BPMN 2 [19] in the beginning of 2011. This specification comes with notable improvements as it combines notation, meta-model, and execution semantics, but it is at the same time completely language independent. This is fine for models meant for documentation. Unfortunately, language independency introduces a semantic gap for executable process models, because they have to be interpreted in – or code generated to – some programming language. But being language independent, information like data-types and input/output parameterization have often to be defined twice for the modeling level and for the technical implementation.

If the adapter layer is generated automatically, this will result in a one to one association, which does not help to abstract from technical details. In the worst case, critical information is omitted, which leads either to round-trip problems or to more information that has to be contributed by the modeler and therefore stored and managed manually. This gets worse the bigger the problem domain is and the more information is omitted. For example most BPM solutions omit information on the (domain-specific) data types or in better cases (e.g. AristaFlow [20]) match some general types like strings and numbers, which leads to a situation where application experts have to deal with compatibility questions between services without any or limited help of the corresponding development environment.

The language and technology independent approach was to achieve substitutionality of implementations of the BPMN 2 standard. As stated in [4], the realizations cover a subset of the standard only and provide proprietary extensions. This ranges from "everything is a task" in jBPM 5 [21] to loosely coupled scripting and technical activities in Activiti [22] losing critical information of the underlying service. Even solutions not bound to the standard BPMN 2 from the scientific context (e.g. AristaFlow) separate the modeling-level from the underlying programming language, resulting in redundant and error-prone definitions. Hence, substitutivity is not achieved although the abovementioned price for the attempt has to be paid. Please note that service-oriented approaches claim to support reusability, but service orchestrations (or business process models) themselves are static entities [23].

[3] Although BPMN and BPMN 2 have the same acronym they should not be confused. The first edition provided a notation only, whereas the second edition is thought as a kind of replacement for both BPEL and BPMN.

3 Second-Order Business Activities

In this paper we extend our approach for creating business process models in a graphical notation in terms of so-called *service logic graphs* (*SLG*) [24] to easily define new variants of a business process simply via second-order parameterization, i.e. via interface-conform exchange of functionality both at design time and at runtime. We

- store (retrieve) service as well as complete process instances in (from) type-safe data objects (*context*),
- bind them to executable nodes in an easy to build and comprehend graphical representation of the process model, and
- execute them as the control-flow reaches the corresponding node.

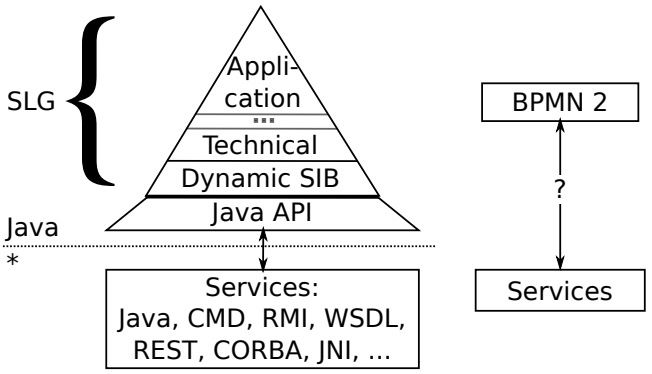

Fig. 1. Dynamic pattern for servification using Java as the base language

Our approach is an evolution of the *extreme model driven development* (*XMDD*) approach [25,26] and its incarnation, the graphical process model design framework *jABC* [27], which already supports domain-specific business activities [28], called *service independent building blocks* (*SIBs*). The enhanced version of SLGs discussed here (jABC 4), bases on the language Java and uses data-types as well as methods of Java objects, integrated on-the-fly into executable graph nodes denoted by *dynamic SIBs* as shown in Fig 1.

Java is platform independent, and provides a lot of adapters to other languages, like e.g. WSDL and REST stubs, which may be generated from the corresponding service descriptions [29], as well as to arbitrary code as far as it is accessible through a standard Java API. Nearly every technology or method can be wrapped into such a Java method as in the *Java Runtime Environment* (*JRE*) there are also interpreters available for a lot of scripting languages, like e.g., Groovy, Jython, and JRuby, and the *Java Native Interface* (*JNI*) allows for accessing platform dependent functions, e.g. via *C* or C++ code. In fact, even command-line tools are accessible via standard Java libraries.

That Java is a good choice for an implementation of such a framework is also reflected by the fact that prominent BPM frameworks, like JBPM 4 and 5, Activiti, AristaFlow, and jABC, all base on Java. However, please note that Java is only used at the 'tool/system-level', e.g. to support consistent typing and other object-oriented features like subtype and parametric polymorphism. The implementation language for the individual processes may indeed be a different language. Moreover all the presented concepts are quite general and could have been implemented in and for other languages like *C++* or *C#*.

We allow for modeling SLGs with business activities in the graphical development environment jABC basing on Java method calls (dynamic SIBs) and domain-specific data structures defined in Java. This leads to a hierarchical graph structure [30], where coarse-grained SLGs classified as 'Application' (level) are successively refined by more technical SLGs indicated by 'Technical', which in turn are based on dynamic SIBs (cf. Fig. 1). The available type information is used to check whether the usage of services and subgraphs is sane. This allows us to narrow the semantic gap and to move the technological break behind a Java API, where only technical experts are involved, and therefore supports the principle "simple for the many, difficult for the few" [31].

Using language features like subtype polymorphism from object-oriented programming is a natural and convenient approach to introduce variability in conventional programming. Similarly, in our second-order process models services and process models are described on the level of their input/output parameterization, which allows us to delay the choice of their concrete implementation until runtime. This enables (runtime) variability on every hierarchy level and not only on the service level (where it is directly supported by Java) without introducing new complication.

Key to this approach is to enhance the usual control-flow-based modeling with explicit data-flow information expressed via type-safe second-order contexts. This increases the comprehensibility of the models and is technically achieved by:

- supporting a type-safe second-order context of data objects storing service as well as process instances,
- equally treating Java methods and subgraphs as executable business activities,
- allowing for dynamic service and process exchange without touching the SLG via providing different service and process instances in the context at runtime in terms of its inner state and its functionality (subtype polymorphism).

The following two sections will stepwisely illustrate our second-order approach to servification. Section 4 models a fictional flight booking web application 'BookAFlight' in a first-order fashion. This modeling is then flexibilized for (runtime) variability by using second-order parameterization in Section 5.

4 A First-Order Solution for a Flight Booking System

This section introduces our example scenario, the process 'BookAFlight', and illustrates its first-order parameter transfer mechanism approach by looking at the activitiy 'one click booking'. The main feature of this illustration is that the user simply initially provides all the required information like his credentials, the payment information, the selected flight etc.. The subsequent booking process, including its payment, is done fully automatically in response to a single click on the button 'one click booking'.

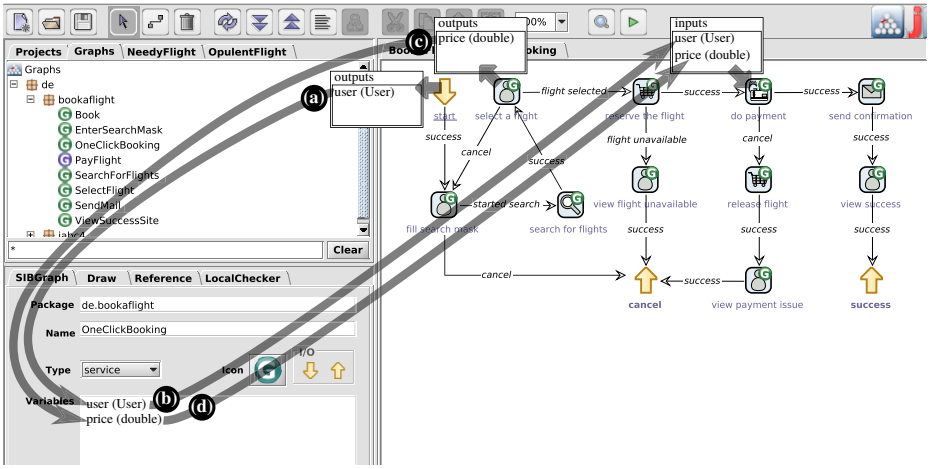

Fig. 2. The main business process 'OneClickBooking', overlayed with the steps necessary for adding the input parameters to a concrete payment service

Fig. 2 shows the jABC4 window and its three main areas, instantiated with the 'OneClickBooking' process model:

1. *browser area*: a tabbed pane containing a *project browser*, a *graph browser*, and one or more *service browsers*. These components offer resource trees which enable to browse e.g. for SLGs and services (Java methods).
2. *inspector area*: a tabbed pane containing several *inspectors*. An inspector gives context information to the currently selected components like nodes in a graph or the graph itself. Some inspectors support editing the information, too. Currently the *graph inspector* is opened, which shows information related to the selected SLG.
3. *graph canvas*: the modeling area, displaying the graph structure of the current SLG and offering functionality for its modification.

We assume, that we have already modeled an SLG for the feature 'one click booking' shown in Fig. 2 using a dedicated payment service. The process then starts with the node labeled 'start' and ends either with the output node 'success'

or 'cancel'. Every inner node is a business activity representing a service or a sub-process execution. The icons give a hint about the nature of an activity. E.g., all nodes with an stylized user icon constitute user interaction.

The parameter transfer between the different activities can be defined simply via drag and drop as illustrated in Fig. 2 for 'OneClickBooking'. In the depicted scenario, the required parameter transfer for 'do payment' is indicated by the four grey arrows:

- Arrow (a): output parameters of a start node are simply the input parameters of the whole SLG. Accordingly, the context variable 'user' functions as a formal input parameter of OneClickBooking.
- Arrow (b) indicates that the value of the context variable 'user', which is used to transfer the according parameter value into OneClickBooking at the node 'start', is required as input for the activity 'do payment'.
- Arrow (c) denotes that the output parameter 'price' of the activity 'select a flight' has to be written to the context variable 'price'.
- Arrow (d) depicts that the value of the context variable 'price', which is written by the activity 'select a flight' is required as input for the activity 'do payment'.

A major feature of jABC4 is the type-safety not only for simple Java types like *double*, but also for domain-specific types like `User`. This enables us to ensure the compatibility of services on the modeling level via the built-in check facility. Compatibility issues like missing or mismatching type information are shown in a dedicated inspector. (Missing) types may be set in the GUI via a dialog for class choosing, which searches in a set of types tailored to the needs of the current domain. After all types have been set, the situation is as shown in Fig. 2. The types are denoted in parentheses behind the labels of context variables, input, and output parameters. Hence, we can be sure that the services are compatible.

5 Generalization to Second-Order Modeling

In this section we show how it is possible to transform our first-order 'one click booking' process to second order. More concretely, we transform 'one click booking', which currently uses one dedicated payment service (cf. activity 'do payment' in Fig. 2), into a context-sensitive second-order process that receives its payment service as a parameter. This requires to exchange the concrete payment activity 'do payment' in the model of 'OneClickBooking' by a second-order activity that obtains its concrete payment method/process at runtime via the context.

Technically, 'do payment' references now 'PayFlight' which is an *interface graph* (cf. Sec. 5.1). In contrast to the usual executable service graphs (like e.g., the whole BookAFlight graph of the previous section), it only defines the abstract input/output parameterization of a payment method precisely enough to establish the required links when retrieving an actual payment method/process at runtime from the context.

In general, second-order activities can be dynamically instantiated by any type-correct instance activities, be they atomic or whole SLGs. In particular, appropriate instance activities may well be transferred as an actual parameter to 'OneClickBooking', or (manually) selected during the process execution, perhaps in the course of an automated service discovery process [32].

The whole transformation for enabling the desired variability, which results in the process depicted in Fig. 3, is described in the following subsections.

5.1 Interface Graphs

The interface graph 'PayFlight' consists of an entry point denoted by the node 'start' and two exit points, namely 'success' and 'cancel', and like 'do payment' it has two input parameters defined, namely the price and a user.

The latter is used to retrieve the payment information[4]. We use a domain-specific type `User` with all necessary information, which may be generated from a UML class diagram. The type can be set in the class chooser.

5.2 Going Second-Order

The transformation for enabling the desired variability as such requires the following three steps, which are illustrated in Fig. 3:

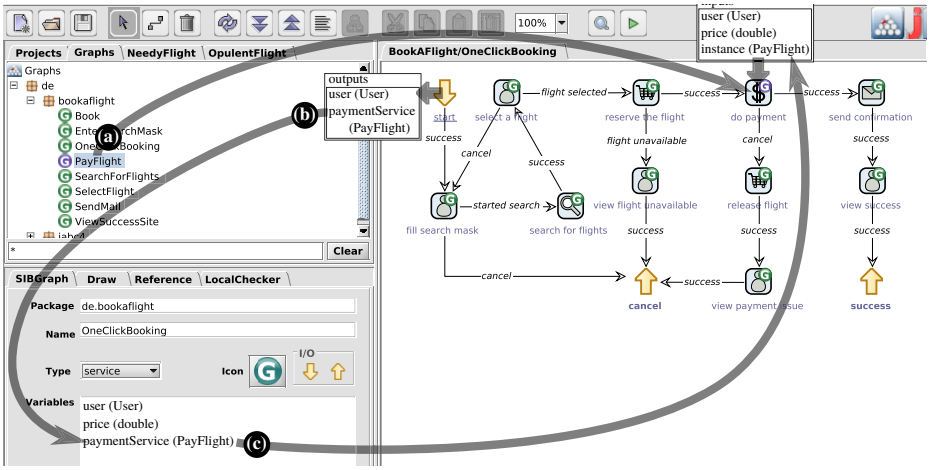

Fig. 3. The main business process 'OneClickBooking' overlayed with the steps necessary for the preparation of automatically integrating external payment services at runtime

[4] E.g., for a payment with credit card, this is the complete credit card information, consisting of full name, credit card number, expiry date, and card verification number.

(a) Exchange the 'do payment' activity by an equally named activity representing the interface graph 'PayFlight' in 'OneClickBooking'. This way the implementation will retrieve its payment method/process from the (execution) context at runtime.
(b) Define an input parameter 'paymentService' at the 'start' activity, implementing the interface (graph) 'PayFlight', and wire it to the context variable 'paymentService' accordingly.
(c) Connect the context variable 'paymentService' as a parameter to the input parameter 'instance' of the activity 'do payment'.

At runtime the concrete payment service will be supplied as input parameter 'paymentService', stored to the identically named context variable, and used as graph instance for execution of the replaced activity 'do payment'.

Please note that here the type of 'paymentService' does not only represent a domain-specific data type like `User`, but constitutes an executable process instance that is guaranteed to implement and respect the interface 'PayFlight'.

5.3 Implementing 'PayFlight'

Up to this point, an application expert has designed a coarse-grained process model for the feature 'one click booking', which takes a process instance realizing a payment method as an argument. What we still lack is the automatic integration of such payment service instances. A corresponding (technical) integration service (graph) has to implement the interface graph 'PayFlight' and therefore has the same input/output parameterization as defined in Sec. 5.1. Such a technical SLG can be constructed by a domain modeler in three steps (cf. Fig. 4):

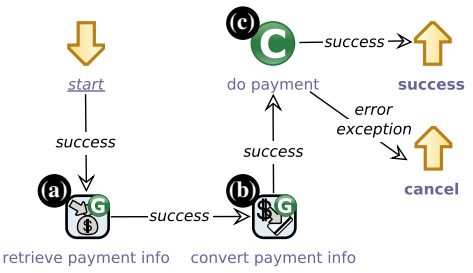

Fig. 4. An example SLG for integrating an external payment service

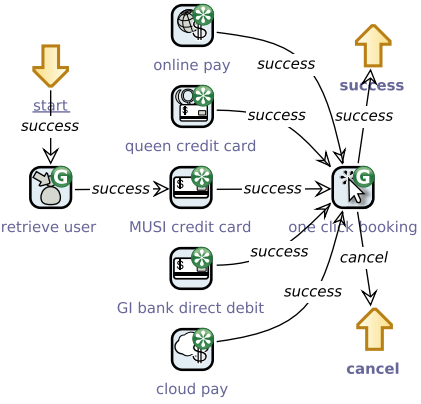

Fig. 5. Example SLG showing how different payment methods may be instantiated and used in the SLG 'OneClickBooking'

(a) Add a business activity for retrieving the payment information for the currently logged in user.
(b) Add a business activity that converts the 'BookAFlight'-specific payment information into data-types of the corresponding payment service.
(c) Wrap the external payment service into a Java method so that it can be used in a dynamic SIB. We will not go into the details of these steps which are typically automated.

In order to prepare a payment service as a second-order parameter for 'OneClickBooking', the context variables have to be connected to the corresponding input and output parameters along the lines (cf. Sec. 4). This can be done independently for each of the candidate payment services, and may also be automated in case appropriate domain information is available (perhaps in form of an ontology).

5.4 Second-Order in Action

In Sec. 5.2 we have shown how to raise 'OneClickBooking' to second-order using an interface graph for integrating different payment services. Let us now show how this can be exploited to exchange these services without touching the process model 'OneClickBooking' at all. Rather, the choice of the service solely depends on the situation/context of 'OneClickBooking', which feeds PayFlight with the correct service via the execution context. Fig. 5 depicts a simple corresponding context process that instantiates a fictional payment service to be transferred to 'OneClickBooking' as a parameter: The small overlay icons in the top right of the *instantiating activities* denote that they instantiate the graph instead of executing it. In this setting, changing the payment service simply means changing the

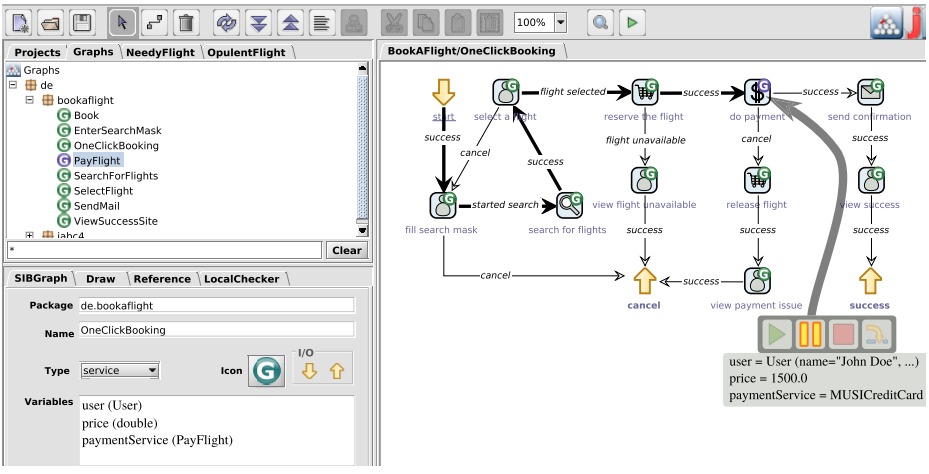

Fig. 6. The main business process 'OneClickBooking', overlayed with the steps necessary for integrating external payment services

edge labeled with 'success' starting in the business activity 'retrieve current user' to another payment services, while leaving 'OneClickBooking' fully unchanged.

Fig. 6 depicts the SLG for 'OneClickBooking' at runtime. The thick control-flow edges denote the history of the actual flow of the current execution, which arrived at the activity 'do payment'. Fig. 6 (bottom right) shows the current status of the context. It reveals that during the execution the payment service 'MUSICreditCard' has been passed as input parameter to 'OneClickBooking' and will be used for transferring $1500.00 for the selected flight.

The process depicted in Fig. 5 is particularly simple. In practice there will be a steadily growing library of payment services and some profile-driven selection mechanism for steering the choice of payment service. In fact, our second-order approach allows that this service library is growing while 'OneClickBooking' is running.

6 Conclusion and Future Work

We have presented second-order servification, a business process modeling paradigm for variability, which is based on considering services and even whole subprocesses as 'resources' of a (second-order) business process, which can be created, selected, and moved around just like data. Its impact, the ease of defining new variants of a business process simply via second-order parameterization, or to exchange its constituent services (and even subprocesses) dynamically at runtime, has been discussed along a simple flight booking scenario. This is particularly useful for long-running processes, as the concrete implementation of a second-order activity in a process model may well be unknown when the process starts, and built-up and exchanged without touching the processes code or model while the process is running. We have successfully used this new approach for modeling the quality assurance process for Springer's Online Conference Service OCS [33].

Being based on formal interface specifications, our approach guarantees the executability of all variants or runtime instantiations fully automatically, and, in particular, without any effort from the business process modelers' side. The required interface conformance is only based on an according servification process for each of the constituent services, which, in general, needs technical expertise. To overcome this bottleneck, we are investigating how far this servification process can be automated for semantically annotated service libraries along the lines indicated by a related project on semantic technologies [34,31].

References

1. Scheer, A.W., Thomas, O., Adam, O.: Process Modeling using Event-Driven Process Chains, pp. 119–145. John Wiley & Sons, Inc. (2005)
2. Scheer, A.W., Schneider, K.: Aris — architecture of integrated information systems. In: Bernus, P., Mertins, K., Schmidt, G. (eds.) Handbook on Architectures of Information Systems, pp. 605–623. Springer, Heidelberg (2006), doi:10.1007/3-540-26661-5_25

3. Margaria, T., Steffen, B.: Service engineering: Linking business and it. IEEE Computer 39(10), 45–55 (2006)
4. Doedt, M., Steffen, B.: An Evaluation of Service Integration Approaches of Business Process Management Systems. In: 2012 35th IEEE Software Engineering Workshop, SEW (2012)
5. Margaria, T., Steffen, B.: Agile IT: Thinking in User-Centric Models. In: Margaria, T., Steffen, B. (eds.) ISoLA 2008. CCIS, vol. 17, pp. 490–502. Springer, Heidelberg (2009)
6. W3C: Web Services Description Language (WSDL) Version 2.0 (2007), http://www.w3.org/TR/2007/REC-wsdl20-20070626/
7. Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C., Langworthy, D., Nadalin, A., et al.: Web services policy 1.2-framework (WS-policy). W3C Member Submission 25 (2006)
8. Karusseit, M., Margaria, T., Willebrandt, H.: Policy expression and checking in xacml, ws-policies, and the jABC. In: TAV-WEB 2008, Proc. Worksh., pp. 20–26. ACM, Seattle (2008)
9. Fielding, R.T.: Architectural styles and the design of network-based software architectures. PhD thesis, AAI9980887 (2000)
10. Pasley, J.: How bpel and soa are changing web services development. IEEE Internet Computing 9(3), 60–67 (2005)
11. White, S.: Introduction to bpmn. IBM Cooperation, 2008–2029 (2004)
12. White, S., Miers, D.: BPMN modeling and reference guide. Future Strategies Inc. (2008)
13. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: On the suitability of bpmn for business process modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 161–176. Springer, Heidelberg (2006)
14. White, S.: Using bpmn to model a bpel process. BPTrends 3(3), 1–18 (2005)
15. Recker, J., Mendling, J.: On the translation between bpmn and bpel: Conceptual mismatch between process modeling languages. In: The 18th CAiSE. Proceedings of Workshops and Doctoral Consortium, pp. 521–532. Namur University Press (2006)
16. Ouyang, C., Van Der Aalst, W., Dumas, M., Ter Hofstede, A.: Translating bpmn to bpel (2006)
17. Ouvans, C., Dumas, M., Ter Hofstede, A., Van Der Aalst, W.: From bpmn process models to bpel web services. In: International Conference on Web Services, ICWS 2006, pp. 285–292. IEEE (2006)
18. Zur Muehlen, M., Recker, J., Indulska, M.: Sometimes less is more: Are process modeling languages overly complex? In: Eleventh International IEEE EDOC Conference Workshop, EDOC 2007, pp. 197–204. IEEE (2007)
19. Allweyer, T.: BPMN 2.0-Business Process Model and Notation. Bod (2009)
20. Dadam, P., et al.: From ADEPT to AristaFlow BPM Suite: A Research Vision Has Become Reality. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 529–531. Springer, Heidelberg (2010)
21. RedHat Software - JBoss: jBPM Website (2012), http://www.jboss.org/jbpm
22. Activiti Team: Activiti BPM Platform (2012), http://www.activiti.org/
23. Margaria, T., Steffen, B., Reitenspieß, M.: Service-oriented design: the roots. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 450–464. Springer, Heidelberg (2005)

24. Steffen, B., Margaria, T.: METAFrame in Practice: Design of Intelligent Network Services. In: Olderog, E.-R., Steffen, B. (eds.) Correct System Design. LNCS, vol. 1710, pp. 390–415. Springer, Heidelberg (1999)
25. Margaria, T., Steffen, B.: Service-orientation: Conquering complexity with xmdd. In: Hinchey, M., Koyle, L. (eds.) Conquering Complexity. Springer (2012)
26. Margaria, T., Steffen, B.: Business process modeling in the jABC: The one-thing approach. In: Handbook of Research on Business Process Modeling, pp. 1–26. IGI Global (2009)
27. Margaria, T., Steffen, B.: Lightweight coarse-grained coordination: a scalable system-level approach. STTT 5(2-3), 107–123 (2004)
28. Steffen, B., Margaria, T., Nagel, R., Jörges, S., Kubczak, C.: Model-Driven Development with the jABC. In: Bin, E., Ziv, A., Ur, S. (eds.) HVC 2006. LNCS, vol. 4383, pp. 92–108. Springer, Heidelberg (2007)
29. Merten, M., Isberner, M., Howar, F., Steffen, B., Margaria, T.: Automated learning setups in automata learning. In: Margaria, T., Steffen, B. (eds.) ISoLA 2012, Part I. LNCS, vol. 7609, pp. 591–607. Springer, Heidelberg (2012)
30. Steffen, B., Margaria, T., Braun, V., Kalt, N.: Hierarchical Service Definition. Annual Review of Communications of the ACM 51, 847–856 (1997)
31. Margaria, T., Steffen, B.: Second-order semantic web. In: 29th Annual IEEE/NASA Software Engineering Workshop, pp. 219–227 (April 2005)
32. Kubczak, C., Margaria, T., Steffen, B., Winkler, C., Hungar, H.: An approach to discovery with miaamics and jABC. In: Petrie, C., Margaria, T., Lausen, H., Zaremba, M. (eds.) Semantic Web Services Challenge. Semantic Web And Beyond, vol. 8, pp. 217–234. Springer, US (2009)
33. Neubauer, J., Steffen, B., Bauer, O., Windmüller, S., Merten, M., Margaria, T., Howar, F.: Automated continuous quality assurance. In: FormSERA. IEEE (2012)
34. Jörges, S., Lamprecht, A.L., Margaria, T., Schaefer, I., Steffen, B.: A Constraint-based Variability Modeling Framework. STTT (2012)

Sustainable Business Models for Services Using Semantic Web Components: Insights from the Field

Mary Tate[1] and Elfi Furtmueller[2]

[1] Victoria University of Wellington, New Zealand
[2] Austrian Science Fund
mary.tate@vuw.ac.nz

Abstract. Semantic web technologies for Human Resource Information Systems (HRIS) are yet to fulfil their full potential. In this paper we reflect on an e-recruiting service development project using semantic web technologies. We use a modified Action Design Research (ADR) lens to organize insights from an innovative, entrepreneurial service offering. The results show that achieving sustainable business models for e-recruiting services based on semantic web components is non-trivial. It requires a rich and continuous interplay between theoretical knowledge (semantic web, service management, and subject-area knowledge); technical knowledge and expertise in building semantic web components; and the community of applicants and recruiters, who use and extend the sematic web components into new service offerings. We further find that the complex, interactive, adaptive, multi-disciplinary and iterative nature of HRIS projects creates challenges in communicating between the stakeholders, and in extracting and presenting theoretical contributions.

Keywords: HRIS, Action Design Research, Semantic Web, Services.

1 Introduction

This study reports on aspects of the design and development of a service-oriented, adaptive and extensible e-recruiting portal using sematic web components, which has achieved a large market share in Europe. The components have been syndicated and extended for specialised market niches (for example medical or IT recruiting). In a previous study, Tate and Furtmueller [1] argued that the Action Design Research (ADR) method [2] required modifications for a service-oriented, multi-organizational context. We use this modified ADR lens to show that continuous exchange between researchers, practitioners and communities is essential to realise the potential of the semantic web to develop sustainable business models based on web services. In the rest of this paper we first offer a brief overview of the ADR lens. We then briefly describe the project showing the interplay of theory, practice and community at each stage, followed by a conclusion.

2 Modified Action Design Research

Action Design Research (ADR) was proposed by Sein et al. [2] as an extension of Design Science [3]. In ADR building the artefact, intervening in the organization, and evaluating

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 26–30, 2013.

the artefact are seen as concurrent processes. While ADR is valuable as a lens for reporting a service development process, some modifications are needed for development of a multi-organization service-oriented ecosystem [1]. The modified ADR method consists of four stages supported by seven principles. The stages include: 1) *problem formulation*, which identifies a research opportunity based on existing and emerging theories and technologies; 2) *building, intervention and evaluation* (BIE), which involves building the artefact, offering it in the community and concurrently evaluating and shaping the artefact based on learning derived from the community context; 3) *reflection and learning*, which involves the ongoing co-creation, adaption and reassembly of the components; and 4) *formalization of learning*, which allows the situated learning from an ADR project to be developed into a general solution for a class of field problems. The ADR method also includes guiding principles and "critical elements". These include: defining the problem as an instance of a class of problems; viewing field problems as knowledge-creation opportunities; ensuring that the artifacts are informed by theory; authentic and concurrent evaluation, which emphasizes that evaluation is continuous and *not* a separate stage that follows building; guided emergence, which aims to capture interplay between structured intervention and organic evolution; and the abstraction and articulation of learning.

3 The Interaction of Theory, Practice and Community

An overview of the inputs and outputs of each ADR stage follows as Appendix 1. The problem formulation stage started by applying theoretical insights to re-conceptualizing e-recruiting, based on the service dominant logic of business (SDL) [4], and academic research in e-HRM e.g. [5], [6] which in turn was informed by engagement with communities of recruiters and job-seekers. The SDL reconceptualises the nature of business exchange. Principles include that the customer is always a co-creator of value, and that the enterprise cannot by itself deliver value, but only offer value-propositions to customers [4]. A "service" has the following characteristics: 1) it identifies or develops core competences of an economic entity that offer competitive advantages; 2) it has the ability to attract potential customers that could benefit from these competences; 3) it facilitates relationships that involve the customers in developing and co-creating customized offerings; and 4) it includes market feedback mechanisms to support continuous improvement. Considering e-HRM systems using an SDL lens, a number of issues and opportunities were identified. Currently there is little motivation for applicants to keep user profiles up to date, which leads to many out-of-date resumes. There is no standardised vocabulary for job titles or skills and many current e-recruiting systems are relatively ineffective at filtering resumes and identifying suitable applicants [6]. As a result, matching is frequently done manually. Instead, recruiting can be seen as a co-creation process involving dialogue between the applicant and the organization, and using the knowledge of both parties to create an offering (the job). E-recruiting services that foster an ongoing co-creation between current and potential candidates and organizations, including: corresponding with applicants about their desired job and their fit to current vacancies; enhancing the playfulness of the interaction; implementing skill competitions; providing the ability for applicants to rank themselves compared to other applicants on the site and regular prompting of

applicants to update their details have been shown to be more effective [6]. Opportunities also exist in unique market niches; and in developing longer-term relationships between applicants and job portals, so that applicants return to e-recruiting portals for ongoing career development.

The next challenge was for semantic web software entrepreneurs Epiqo [1] to develop a sustainable business model. Research and practice suggested that semantic web technologies would be suitable for the type of service that was envisaged. "S*emantic technologies… have untapped potential for dynamic customization/composition of services*" [7]. While many recruiting functions are common across organizations, the vocabulary, skills and qualifications associated with specific niches varies. However, there have been issues with scaling semantic web services '*the central deficiency of the Semantic Web is their static model of knowledge (ontologies), which implies static and predefined meaning of web-content"*[8]. Therefore, a solution that drew on the resources of the community to maintain the ontology was required. A collaboration between researchers and practitioners in service management, e-HRM and semantic web resulted in a conceptual design for a number of components.1) An ontology for HR sourcing and digital resume design. 2) A workflow process for ontology extension supported by a web crawler. New terms were to be placed in a grey-list for classification. Following initial classification of grey-list entries, the community (i.e. applicants, recruiters, developers) would be "crowd-sourced" to confirm the categorization of unknown terms.[2] This allows the ontology to be learning, adaptive, extensible, and quality assured by its user community. 3) Adaptive and context sensitive interfaces. 4) An adaptive and automated matching process based on the ontology.

The BIE stage primarily involved building the components; ensuring that the components were adapting to the environment; that learning and extension processes worked; and that the service offerings were adaptive to specific niches.

The reflection and learning stage exemplifies the guided emergence principles of the ADR method. Unlike a project in an intra-organizational context, these services are continuously interacting with, and being shaped by, their communities.

Finally, we found that formalising learning from projects of this nature is challenging, as the complex interplay of theory and practice from multiple disciplines means that specific insights for individual disciplines are difficult to extract and contextualise.

4 Conclusion

New business models such as the SDL are converging with semantic web technologies. However, developing sustainable businesses requires rich insights from multiple academic disciplines, practitioners and stakeholder communities in the business domain. In particular, the inflexible nature of ontologies must be addressed in order to provide compelling service value-propositions that can be adapted, reassembled and extended. This project shows that theoretical knowledge, technical competence, business competence, and the domain expertise of stakeholders alone are

[1] `http://epiqo.com/en`

[2] Variations on this approach are sometimes referred to as the "Pragmatic Web".

not sufficient for developing successful semantic web-based products and services. All of these must work in close and continuous co-operation.

Appendix

Table 1. The interaction of theory, practice and community

Inputs	Modified ADR phase/principles	Outputs
Theory 1. Service management, service dominant logic of business 2. E-HRM issues and challenges 3. Semantic web issues and challenges Practice 1. Semantic web entrepreneurs 2. Semantic web developers 3. HR professionals 4. E-HRM portal providers Community 1. Job applicants 2. Developers 3. Organizations 4. Niche professional communities	Problem formulation, 1. Identifies and conceptualizes a research opportunity based on existing and emerging theories and technologies 2. Defining the problem as an instance of a class of problems 3. Viewing field problems as knowledge-creation opportunities; 4. Ensuring that the artefacts are informed by theory.	1. Generic e-recruiting limitations identified from research in theory and practice: poor matching of jobs to applicants out of date resumes; reliance on manual processes. 2. Opportunities identified: improved matching, based on and extensible and learning ontology; improved adaptive and context sensitive online interaction between applicants and recruiting sites; establishment of a long term relationship between applicant and site; added value features for career management 3. Challenges addressed: maintenance and extension of ontology "crowd-sourced" using pragmatic web principles; workflow process for ontology extension. 4. Sustainable business models developed: reassemble/extend components into new and unique niche offerings.
Practice 1. Semantic web entrepreneurs 2. Semantic web developers 3. HR professionals 4. E-HRM portal providers Community 1. Job applicants 2. Developers 3. Organizations 4. Niche professional communities (e.g. chemical engineers)	Building, intervention and evaluation (BIE) 1. Building the artefact, intervening in the community and concurrently evaluating and shaping the artefact based on learning derived from the community context 2. Authentic and concurrent evaluation emphasizes that evaluation is continuous and *not* a separate stage that follows building.	1. Semantic web components 2. Identifying the learning and shaping required and automating it. 3. Business process and components for ontology maintenance and extension and automated job-applicant matching 4. Entrepreneurial service offerings to the community: Components and business models adopted, adapted, reassembled and extended (e.g. niche-portal franchises).

Table 1. (*Continued.*)

Practice and community 1. Use, uptake and market-share of semantic web components, e.g. intelligent matching and intelligent and learning ontology 2. Automated and real-world conversations inform extension	Reflection and learning 1. Ongoing co-creation, adoption, adaption and reassembly of the components; 2. Guided emergence: the interplay between structured intervention and organic evolution.	1. Continuous learning and extension of ontology 2. Extension and recombination of semantic web components into unique business propositions 3. Increased market share
Researchers, practitioners and community 1. Experience	Formalization of learning 1. Situated learning from the project is further developed into a general solution for a class of field problems.	1. Experience with the project informs researchers, practitioners and the community. However, the complex, inter-related, and iterative nature of the inputs makes the learning difficult to report truthfully.

References

1. Tate, M., Furtmueller, E.: Service Development as Action Design Research: Reporting on a Servitized E-Recruiting Portal. In: Proceedings of SIGSVC Workshop. Association for Information Systems, Sprouts Working Papers on Information Systems, vol. 12 (2012)
2. Sein, M., Henfridsson, O., Purao, S., Rossi, M., Lindgren, R.: Action Design Research. MIS Quarterly 35, 37–50 (2011)
3. Hevner, A., March, S., Park, J.: Design Science in Information Systems Research. MIS Quarterly 28, 75–105 (2004)
4. Vargo, S.L., Lusch, R.F.: Evolving to a New Dominant Logic for Marketing. Journal of Marketing 68, 11–17 (2004)
5. Tate, M., Furtmueller, E., Wilderom, C.: Localizing versus standardizing electronic human resource management: Complexities and tensions between HRM and IT departments. European Journal of International Management (in press)
6. Furtmueller, E., Wilderom, C., Tate, M.: Managing Recruitment and Selection in the Digital Age: E-Hrm and Resumes. Human Systems Management 30, 243–259 (2011)
7. Janev, V., Vranes, S.: Applicability Assessment of Semantic Web Technologies. Information Processing and Management 47, 507–517 (2011)
8. Pohjola, P.: The Pragmatic Web: Some Key Issues. In: I-SEMANTICS ACM International Conference Proceedings Series (2010)

Measuring Sales Cannibalization in Information Technology Markets: Conceptual Foundations and Research Issues

Francesco Novelli

SAP Research Darmstadt, Bleichstrasse 8, 64283, Darmstadt, Germany
francesco.novelli@sap.com

Abstract. Sales cannibalization – i.e., intra-organizational sales diversion – bears a prominent role in the competitive upheavals within Information Technology markets. However, detection and measurement thereof have only raised lukewarm interest among Information Systems scholars so far. To their defense, relevant methodological contributions are scattered across several disciplines, base themselves on equivocal definitions of cannibalization, present an overwhelming range of model specifications, and overlap with research on product and technology substitution. Therefore, we provide an interdisciplinary review of the literature on cannibalization, formulate a novel, clear-cut definition of the phenomenon, and clarify its relationship with substitution. Our other contributions are an exhaustive list of the modeling requirements necessary to describe the phenomenon, a compendium of cannibalization measurement models, and a summary of the findings with regard to Information Technology artifacts. This work should provide an adequate foundation and identify promising topics of study for further research endeavors in this domain.

Keywords: cannibalization, substitution, empirical models, literature review.

1 Introduction

We judge this very moment propitious for Information Systems Research (ISR) on sales cannibalization – the phenomenon of intra-organizational sales diversion – given its crucial role in Information Technology (IT) markets. An exemplar occurrence is the recent launch of the Apple iPad and the subsequent "gold rush" into the tablet market by manufacturers of personal computers and smartphones alike. Since tablets appeal to potential buyers of their other established product lines, those manufacturers offer the flank to sales cannibalization [1].

The cannibalistic threat may also move upstream and downstream along the value chain, affecting platform product vendors and complementors. Sales cannibalization affecting PC manufacturers, for example, is a key factor for their chip suppliers [2]. In addition, the adoption of tablets and other content-consuming devices is germane to the more or less cannibalistic way publishers make content which they already offer through traditional publishing channels available for consumption through such devices.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 31–42, 2013.

Cannibalization patterns and their magnitude are often sources of controversy. Uncertainty as to whether and to which degree a firm's product portfolio is affected by cannibalization may lead to an unclear perception of its expected performance by both internal and external stakeholders. Conversely, a reliable quantification of this phenomenon would improve understanding and transparency with regard to a strategic facet of competition in Information Technology (IT) markets.

With these topical issues in mind, sales cannibalization represents a fruitful area of investigation in ISR, with an evident managerial relevance for businesses operating in today's IT markets. However, progress in the understanding of this phenomenon is hampered by the scattered state of contributions from multiple disciplines (alongside ISR, mostly Marketing Science and Management Science). Therefore, our goals are to synthesize the interdisciplinary literature on this topic and uncover gaps that could reveal novel research streams. For researchers, this work may be a foundation for further investigations, responding to the call of Webster and Watson for review articles that would strengthen ISR as a field of study and accelerate its accumulation of knowledge [3]. This is particularly relevant, we believe, for no other literature review on cannibalization exists. For practitioners, we will provide a compendium of the available measurement methodologies and a synopsis of relevant findings.

The structure of the paper is as follows: we start by revisiting the generic definition of sales cannibalization and its relationship with the concept of substitution (section 2). We then review the available empirical approaches for measuring cannibalization (section 3). In the following section (4) we summarize the specificities of the phenomenon in the context of ISR and the findings of previous measurement studies. Finally, promising future directions for ISR in this area are sketched out (section 5), before we present our concluding remarks (section 6).

2 Sales Cannibalization: An Overview and Basic Concepts

An accurate, comprehensive definition of sales cannibalization is a strict requirement for an accurate and comprehensive literature review. Moreover, the absence of a generally accepted definition is a shortcoming acknowledged by marketing researchers themselves [4]. Table 1 lists the most frequently cited cannibalization definitions from the marketing literature. Three common constituent parts are evident: the economic entities whose generated sales benefit or suffer from the occurring of the phenomenon – from now on respectively "cannibal" and "victim" –, the common organizational realm their revenues accrue to, and the specific relationship which connects their sales-generating processes.

The cannibal and victim entities are products or services, or sets of products or services whose sales are aggregated along some dimension (e.g., the channel through which they are distributed). One possible caveat lies in the a priori identification of a distinguishing trait between the two entities. Ref. [5] associates cannibal and victim with a *new* and an *old* product, and there is indeed a rich body of research along these lines [4, 6–8]. However, other scholars either distinguish cannibal and victim along a different dimension [9, 10] or detect the phenomenon's patterns and direction endogenously in the analysis (without any a priori identification) [11, 12].

Table 1. Most frequently cited definitions of sales cannibalization

Source	Definition
[5]	*The process by which a new product gains a portion of its sales by diverting them from an existing product.*
[13]	*'Redistributed' revenue, in that existing buyers are substituting one item for another in the company's product portfolio.*
[14]	*Competition within a firm's own product line.*
[11]	*The extent to which one product's customers are at the expense of other products offered by the same firm.*

The second cardinal element in the definition of cannibalization is the organizational realm within which the phenomenon can be properly called *cannibalization* as opposed to *competitive draw*. The former's perimeter is identified by the boundaries of the organization benefiting from the cumulative sales of cannibal and victim. As a matter of fact, all of the authors mentioned in Table 1 defined cannibalization explicitly as an intra-firm phenomenon.[1] The firm of reference can be the manufacturer/provider or any intermediary (e.g., a retailer, as in [15]).

The most critical definition element is the relationship between the sales-generating processes of the cannibal and victim entities, i.e., what to "gain", "divert", "redistribute", or "substitute" sales precisely mean. Cannibalistic patterns are properly detected by comparing the actual purchase decision with the hypothetical one which would have been taken in absence of the cannibal item. In other words, those among the cannibal's customers who would have bought the victim, had the cannibal not been in their available choice set, are qualified as cannibalized. This aspect is instrumental in assessing the validity of the measurement approaches proposed in the literature for, given the difficulty of eliciting or recording customers' intentions, most models can only estimate these cannibalistic buying patterns from historical purchase data.

In conclusion, we propose to define cannibalization as the intra-organizational phenomenon of sales diversion by means of which sales of a product or service (the cannibal) are generated by diverting potential sales that a substitute product or service (the victim) would have obtained in absence of the former, *ceteris paribus*, within a common organizational realm collecting the revenues of both.

The question could be posed as to whether cannibalization represents merely an emphatic synonym for product substitution and, therefore, we will briefly dwell on the relationship between the two concepts. Just like cannibalization, substitution is a multidisciplinary topic that has been drawing attention from several fields of investigation. In his seminal work on strategic management Porter defines substitutes as "products that can perform the same function" [18, p. 23], distinguishing between closer substitutes by competitors within the same industry and more distant ones by latent competitors outside of the industry boundaries. The industry itself is defined by an arbitrarily chosen level of substitutability to distinguish the two competitive realms

[1] Although Heskett does not constrain cannibalization this way in his definition, he only mentions examples where that is the case [5, pp. 115-118 and 150-152], so it might have been merely an oversight not to state it explicitly.

of "direct rivalry" and "threat of substitution" [id., p. 32]. Based on our definition, even though the cannibal and victim entities must exhibit some degree of substitutability to engender the interdependent buying patterns described above, their revenues must accrue to the same organizational realm.

In microeconomics two goods are called substitutes if a price change of one of them has an effect of equal sign on the demand for the other. Formally, the cross-price elasticity (or cross-elasticity) of demand for the latter with respect to the price of the former must be positive [19, p. 52]. The antitrust literature provides a *trait d'union* between the microeconomic and the strategic perspectives: renowned antitrust cases namely relied on the cross-elasticity of demand, as a proxy for product substitutability, to delineate the relevant market boundary [18]. Given, as already mentioned, that cannibal and victim are substitute products, cross-price elasticities have a central role in some of the empirical approaches we will mention below.

Substitutability, investigated in the context of consumer choice theory, represents a "negative similarity effect" [19], whereby a new item will take relatively more share away from items similar to it than to dissimilar ones, that is, disproportionately compared to predictions based on the principle of independence of irrelevant alternatives (IIA). Experimental and analytical approaches have been developed to derive brands' differential substitutability from physical or perceived attributes and to predict market share movements accordingly. Some cannibalization measurement approaches are indeed rooted in choice theory, either benchmarking actual market shares changes against IIA predictions [4] or incorporating choice models [20].

When market penetration encompasses multiple technology generations with dependencies among their diffusion processes, *technological substitution* represents a phenomenon of interest in diffusion research (see [21] for a recent overview). In this domain, substitution is the mechanism by which adopters and potential adopters of preceding generations of a base technology opt for a successive generation [22]. Of course, substitution may be a driver of intergenerational cannibalization whenever the same firm simultaneously offers products relying on distinct technology generations. In such a scenario, multigenerational diffusion models have been employed to measure cannibalization [23, 24].

Notwithstanding the interdependences between the concepts of substitution and cannibalization we have outlined so far, an important distinction must be stressed. We defined "cannibalistic" as the buying patterns whereby the customers who *would have bought* the victim, had the cannibal not been in the available choice set, would have purchased the cannibal (columns *a*, *c* and *d* in Table 2). However, the process of substitution encompasses both these customers and those who *bought* the victim and then switched or upgraded to the cannibal (columns *a*, *b* and *e* in Table 2). Therefore, while cannibalization only considers the victim's *potential* customers, substitution considers both *potential* and *former* customers. The magnitude of the two phenomena may greatly differ depending on the number of consumers whose buying pattern adheres to each profile.

In conclusion, the occurrence of cannibalization implies a positive degree of substitutability between cannibal and victim, and cannibalization can be regarded as *one component* of an intra-organizational substitution process. From a methodological point of view, research on cannibalization measurement was greatly enhanced by the research streams on substitution we have briefly recalled here.

Table 2. A comparison of patterns of cannibalization and substitution

	(a)	(b)	(c)	(d)	(e)
	Actual purchase decision	*Prior* purchase decision	*Hypothetical* purchase decision with a choice set without cannibal	Qualified as cannibalization	Qualified as substitution
I	Cannibal	Victim	Victim	✓	✓
II			Competitor		✓
III			Leave the market		✓
IV		Competitor	Victim	✓	✓
V		Non-consumer in the category		✓	✓

3 Detection and Measurement of Sales Cannibalization

Detection and measurement of sales cannibalization are the primary goals of empirical studies which propose methodologies to reveal and quantify the phenomenon. From a descriptive point of view, the nature of the cannibalization phenomenon is fully determined if we are able to specify it in terms of patterns, magnitude and variation over time. This translates into a series of modeling requirements that we have collected from the literature and listed in Table 3. However, given space constraints, only a few requirements will be discussed here.

The analysis of cannibalization patterns should take into consideration the possibility of asymmetries [12] and the involvement of items both within and between product categories [8]. When the study does not target a predefined pair of entities (e.g., the new and old product pair), the model should be able to account for multivariate cannibalization within a given set of products, that is, support the discovery of which item is diverting customers from which other item. With regard to the temporal dimension, cannibalization may change over time due to customers' heterogeneity in adoption timing and other disturbances [8]. It may also produce cross-period effects [25] and alter the long-term performance of the victim [26].

The array of measurement models in the literature can be subdivided into descriptive models, where a response model is not explicitly formalized, and econometric models, which mathematically formalize the sales response of one or more product items, feeding it with time-series data in order to estimate cannibalization. The former are less demanding in terms of data requirements and analytical complexity, but *alert* us of a possible cannibalization issue rather than providing actual measurements. Among the studies which propose such models, one finds an ecology-inspired purchasers' cluster analysis to estimate cannibalization *potential* among brand's variants [11]. Relevant customers' characteristics are used to define a multi-dimensional space in which each variant has a niche, i.e., a perimeter containing its customers within a certain distance from its average buyer. A measure of overlap between the variants' niches then serves as a proxy for cannibalization potential.

Lomax and her co-authors proposed three techniques to detect cannibalization engendered by a new product launch: gains loss analysis, duplication of purchase analysis, and share movements analysis [4]. The first technique consists in taking the ratio of the victim's market share loss to the cannibal's market share gain (see also [27]). The higher the ratio compared to competitors' brands, the more plausible a cannibalistic explanation of the cannibal's gains in the market. Duplication of

purchase analysis compares empirically-measured levels of cross-purchasing with the predictions made according to the independence of irrelevant alternatives (IIA) assumption. Similarly, share movements analysis benchmarks the actual change between pre- and post-launch shares of purchase against the IIA-predicted ones. Deviations from the expected values signal potential cannibalization in both cases.

Table 3. Modeling requirements identified in the literature (in chronological order)

Modelling requirement		Description	Source
I	Asymmetry of cannibalization patterns	Sales of one product mix item may affect sales of a second item differently than the other way around.	[28]
II	Variation of cannibalization patterns over time	Diversion of sales among two items in the product mix may change over time (in magnitude and/or direction).	[28]
III	Multivariate cannibalization	A cannibal may divert sales from multiple victims. Conversely, a victim may lose sales to multiple cannibals.	[6, 11, 12]
IV	Stochastic effects	The cannibalization patterns may be subject to temporary nondeterministic disturbances.	[6]
V	Long term effects	The addition of the cannibal may change the underlying (base) sales-generating processes.	[26]
VI	Cross-period effects	Cannibalistic shifts in sales may encompass stockpiling or anticipation and therefore produce lead or lagged effects.	[15] (lagged effects only)
VII	Cross-sectional heterogeneity	Sales response may differ depending on the considered aggregate data cross-section (e.g., store).	[7]
VIII	Customers heterogeneity	Potential customers may react differently to the presence of the cannibal, in terms of the type of response or its timing.	[29]
IX	Inter-category effects	Cannibalistic sales diversion may take place also among items which belong to different product categories.	[8]

Econometric models provide a mathematical formalization of the sales-generating process for one or more of the entities. These models differ in the way they tackle the cannibalization measuring problem. A first group of models specify a sales response function for the victim entity alone and devote some explanatory variables to formalize the impact of the cannibal, detecting any reduction in the victim sales and attempting to explain it in terms of the cannibal's introduction, presence, and/or attributes. This approach is employed by scholars assessing the cannibalization effect of a companion website for a printed publication. In ref. [26], for instance, a structural-break unit-root test is employed to verify whether the web companion negatively affected the circulation and advertising revenues of national newspapers. In that study, the only required information about the cannibal is the website's launch date. In ref. [30] a cannibal's attribute is integrated into the model, namely the degree of overlap with the printed content. When analyzing the cannibalization of retail-stores revenues by the online channel in [15], not only is the existence of the cannibal considered but the cannibal-generated monetary sales as well. All of the cannibalization studies relying on Amazon rankings as proxies for sales volume [10, 31–34] also formalize the victim's response function alone.

Some scholars take a specular approach, decomposing the cannibal's demand to ascertain whether diversion of sales from other items in the firm's portfolio lies among the cannibal's sources of demand. In ref. [6] a dummy variable regression is

employed and a process equation accounts for the time-varying nature of the cannibalization effect. In ref. [8] a vector error-correction model allows linear unit-sales decomposition as a method to study a radical innovation diverting customers from other product categories. Multigenerational diffusion models that only support forward substitution also fall into this category [23, 24].

Another body of research approaches the entities symmetrically from the modeling point of view. That is the case for multigenerational diffusion models which contemplate both forward and backward substitution [28], and for the response models conceived as systems of log-linear equations [12, 35, 36]. The latter set of models does not allow the quantification of cannibalization in absolute terms but rather through cross-elasticities, which may eventually be turned into sales diversion ratios [36].

Finally, some researchers have quantified the cannibalization effect by first calibrating a discrete choice model and then simulating a hypothetical scenario where the cannibal item is removed from the choice set. The appropriately computed delta in sales of the victim constitutes cannibalization [20, 37]. Conjoint analysis can also be used in an analogous fashion [38].

4 Sales Cannibalization in Information Systems Research

In this section we will review the findings of prior measurement studies specific to cannibalization occurrences in IT-related markets. Sales cannibalization represents a ubiquitous phenomenon in markets related to information goods and information systems. Indeed, a tally of *alleged* casualties may be compiled: traditional media (records, books, newspapers, television broadcasts, etc.) cannibalized by their digital counterparts, packaged software cannibalized by software-as-a-service, enterprise servers cannibalized by cloud computing, and traditional advertising cannibalized by online ads – just to name a few recent occurrences in the press.

With regard to information goods, the Internet has played a central role in determining cannibalistic situations for content providers. Successive generations of online platforms have namely allowed content providers and content consumers to transact in an ever increasing range of formats and channels (without generating too much enthusiasm on the supply side, to couch it euphemistically). Table 4 details the array of possible buying situations resulting from such a platform evolution, based on the nature of the purchased entity (physical good, logical good, service) and on the type of underlying platform (retail or e-commerce).

The Internet has begot a first cannibalistic situation by providing an alternative channel to sell information goods in physical form that were already being distributed through retail stores [15]. Following the widespread adoption of portable media players and under the pressure of piracy, a new generation of online stores has then arisen, where the same information goods can be purchased as individually downloadable encoded files [32, 39]. The most recent development is the shift towards a service paradigm, where users can access digital content on-demand through dedicated service providers, such as Amazon, NetFix, or Spotify [34].

Table 4. Information goods' buying situations

	Brick & Mortar platform	Internet-based platform
Information good with physical manifestation	On paper; on disc	On paper; on disc
Information good with purely logical manifestation	Prepaid gift card	Download of an encoded file
Service	Exhibition; performance; broadcast	Hosted data services; on-demand services

Most IT vendors are also vulnerable to cannibalization. If we classify the technological components of information systems into the four product families of hardware, software, databases, and telecommunications [40], we can readily see cannibalistic situations in all of them. For instance, chip manufacturers currently see cannibalization both among microprocessors for enterprise servers and those designed for data centers, and among low-consumption microprocessors for mobile devices and the more powerful ones for personal computers; database vendors witness the same phenomenon arising between in-memory and traditional database offerings; network operators among traditional phone services and VoIP. With regard to software, higher than average cannibalization rates and the ability to successfully introduce a new product during the growth phase of the previous one have been found to be distinguishing features of successful software vendors [41].

Several empirical studies have estimated cannibalization rates in IT-related markets, whereby the great majority have focused on information goods and only few articles have dealt with other types of IT products and services. Existing measurement studies on cannibalization of information goods are enumerated in Table 5. Most scholars address the cannibalizing impact of digital, electronically distributed media on sales of their physical counterparts (printed publication or CD/DVD). The exceptions are ref. [15], an assessment of the cannibalization problem for a retailer selling physical copies both online and through its stores, and references [31] and [10], investigating how the introduction of online secondary markets affected sales of brand-new copies.

The findings of these studies are detailed by market segment in Table 6. The results with regard to the press market are partly inconclusive, as detected cannibalization rates (in terms of reduced circulation of the printed edition) ranged from insignificant [26] to noteworthy [42]. In ref [29] the effects of customers' heterogeneity are revealed, showing that cannibalization rates differed from one age group to the other. This result was confirmed in the market for academic publications, where some customers see the printed version and the PDF one as substitutes, others as complements [39]. Content markets segmentation was also revealed in [32]: legitimate digital and physical copies of NBC television series were not seen as substitutes by most customers.

Outside of the markets for digital content, few scholars have attempted to measure sales cannibalization. In ref. [23] a multigenerational diffusion model is used to estimate cannibalization rates (as the percentage of technology adopters buying the latest available generation) for successive generations of IBM mainframe computers. The estimated rate ranges from 90% for the second generation to 34.5% for the fourth one. In another multigenerational diffusion study, this time in the market for game

consoles, [24] estimated that less than 10% of the customers of the Sony Playstation-2 were cannibalized from potential Playstation-1 adopters. In ref. [9] field auctions are employed to experimentally assess the cannibalization potential of remanufactured products, auctioning a network security device by CISCO in brand-new and remanufactured form. Based on the analysis of bidding behaviors and bid results, the hypothesis of cannibalization was rejected.

Table 5. Measurement studies on cannibalization in markets for information goods

Source	Product category	Product forms (Cannibal / Victim)	Horizon
[26]	Newspapers	Online edition / printed edition	1990-2001
[15]	Music records	Online sales / retail sales	1998-1999
[42]	Newspapers	Online edition / printed edition	1976-2001
[31]	Books	Used copy / new copy	2002-2004
[20]	Newspapers	Online edition / printed edition	2000-2003
[29]	Magazines	Online edition / printed edition	1996-2004
[30]	Magazines	Online edition / printed edition	1996-2001
[10]	Music records; movies	Used copy / new copy	2004
[43]	Movies	Free television broadcast / DVD	2005-2006
[39]	Academic publications	Digital purchase / Printed book	2002-2004
[32]	TV programs	Digital purchase / DVD	2007-2008
[34]	Movies	Digital rental or purchase / DVD	2008

Table 6. Comparative review of findings on information goods cannibalization

Source	Cannibalization[a]	Empirical Approach
Market segment: press (newspapers and magazines)		
[26]	No significant cannibalization	Structural-break
[42]	–3.1% (short-term) –26.4% (long-term)	Discrete choice modelling (aggregate logit)
[20]	–1.47%	Discrete choice modelling
[29]	–4.2%	Discrete choice modelling (nested logit)
[30]	–3-4%	Fixed-effects
Market segment: academic press		
[39]	–2.44% (short-term) +10% (long-term)	Structural-break
Market segment: entertainment (music and video)		
[15]	2.80% [b]	Simultaneous dynamic equations
[32]	No significant cannibalization	Difference-in-difference
[34]	–41.6%	Fixed-effects
Market segment: secondary markets		
[31]	16% [c] (books)	Discrete choice modelling (aggregate logit)
[10]	24% [c] (CDs), 86% [c] (DVDs)	Discrete choice modelling (aggregate logit)

a) Percentage change in victim's unit sales due to cannibalization, unless otherwise noted. b) Percentage of cannibal's monetary sales diverted from the victim. c) Percentage of cannibal's unit sales diverted from the victim.

5 Research Issues

Our review of the literature uncovered a gap between the rampant role of sales cannibalization in IT markets and the related empirical work in ISR. Thus, we will briefly sketch some research themes that, in our view, would help to bridge that gap.

Since the Marketing Science field has supplied the majority of cannibalization measurement methodologies, but applied them mainly to consumer packaged goods,

one immediate need for ISR scholars would be to conduct replication studies which test the proposed model specifications on IT goods and services.

With regard to specific ISR subdomains, we found that existing studies predominantly focus on information goods. Cannibalization in other IT areas has barely been touched, although some IT segments are experiencing dematerialization and servitization trends analogous to those illustrated for information goods. Special-purpose devices are being substituted for functionally-equivalent software applications on general-purpose computers, often offered by the same vendor. Manufacturers of personal navigation devices, for instance, have started developing navigation software for GPS-enabled smartphones [44]. Software sales and distribution channels are increasingly digitalized as well, even in enterprise software markets [45]. Hardware and software resources are increasingly delivered as services, labeled as cloud computing, software-as-a-service, on-demand, etc. This servitization trend poses additional challenges due to the different revenue models that victim and cannibal may employ (e.g., perpetual licensing vs. subscription). In such a scenario, cannibalization quantified in terms of monetary sales may be more meaningful than in terms of unit sales as common in the marketing literature. Table 7 recapitulates these trends and some illustrative cannibalistic situations they beget for IT vendors.

Table 7. Trends engendering sales cannibalization in IT-related markets

	Information goods	**IT products and services**
Dematerialization	Physical manifestation *vs*. purely logical manifestation.	Special purpose devices *vs*. software applications on general purpose devices; Online *vs*. traditional channels for software sales & distribution.
Servitization	Discrete purchases *vs*. on-demand services.	Enterprise servers *vs*. cloud computing; On-premises applications *vs*. software-as-a-service.

6 Summary and Conclusions

We reviewed the interdisciplinary literature on sales cannibalization from the ISR perspective to provide the foundations for further research in an area we deem of topical interest for scholars and practitioners alike. A revised and precise definition of the phenomenon was given, clarifying its befuddling relationship to the concept of substitution. Moreover, we presented an exhaustive collection of the requirements a modeling endeavor should meet to describe the phenomenon adequately, followed by a compendium of the measurement methodologies proposed in the literature. Findings on the cannibalization rates experienced by information goods and IT purveyors were reviewed as well. Finally, we suggested some pertinent directions for future research.

Acknowledgements. This work was partially financed by the European Commission under grant agreement 285248 (project FI-WARE).

References

1. Bustillo, M.: Retailers Turn to Gadgets, http://online.wsj.com/article/SB10001424052748703376504575491533125103528.html
2. Merritt, R.: Otellini says Intel to take tablet market, http://www.eetasia.com/ART_8800623147_1034362_NT_4c6b27ab.HTM
3. Webster, J., Watson, R.T.: Analyzing the Past to Prepare for the Future: Writing a Literature Review. MIS Quarterly 26, xiii–xxiii (2002)
4. Lomax, W., Hammond, K., East, R., Clemente, M.: The measurement of cannibalization. Journal of Product and Brand Management 6, 27–39 (1997)
5. Heskett, J.L.: Marketing. Macmillan, New York (1976)
6. Reddy, S.K., Holak, S.L., Bhat, S.: To extend or not to extend: Success determinants of line extensions. Journal of Marketing Research XXXI, 243–262 (1994)
7. Van Heerde, H.J., Mela, C.F., Manchanda, P.: The Dynamic Effect of Innovation on Market Structure. Journal of Marketing Research XLI, 166–183 (2004)
8. Van Heerde, H.J., Srinivasan, S., Dekimpe, M.G.: Estimating Cannibalization Rates for Pioneering Innovations. Marketing Science 29, 1024–1039 (2010)
9. Guide, V.D.R.J., Li, J.: The Potential for Cannibalization of New Products Sales by Remanufactured Products. Decision Sciences 41, 547–572 (2010)
10. Smith, M., Telang, R.: Internet exchanges for used digital goods (2008)
11. Mason, C., Milne, G.: An approach for identifying cannibalization within product line extensions and multi-brand strategies. Journal of Business Research 31, 163–170 (1994)
12. Carpenter, G.S., Hanssens, D.M.: Market expansion, cannibalization, and international airline pricing strategy. International Journal of Forecasting 10, 313–326 (1994)
13. Kerin, R., Harvey, M., Rothe, J.: Cannibalism and new product development. Business Horizons 21, 25–31 (1978)
14. Moorthy, K.S.: Market segmentation, self-selection, and product line design. Marketing Science 3, 288–307 (1984)
15. Biyalogorsky, E., Naik, P.: Clicks and mortar: the effect of on-line activities on off-line sales. Marketing Letters 14, 21–32 (2003)
16. Porter, M.E.: Competitive strategy: Techniques for analyzing industries and competitors: With a new introduction. Free Press (1980)
17. Bain, J.S.: Price theory. Holt (1952)
18. Werden, G.J.: The History of Antitrust Market Delineation. Marquette Law Review 76, 123–215 (1992)
19. Huber, J., Puto, C.: Market boundaries and product choice: Illustrating attraction and substitution effects. The Journal of Consumer Research 10, 31–44 (1983)
20. Gentzkow, M.: Valuing new goods in a model with complementarities: Online newspapers (2006)
21. Peres, R., Muller, E., Mahajan, V.: Innovation diffusion and new product growth models: A critical review and research directions. International Journal of Research in Marketing 27, 91–106 (2010)
22. Norton, J.A., Bass, F.M.: A diffusion theory model of adoption and substitution for successive generations of high-technology products. Management Science 33, 1069–1086 (1987)
23. Mahajan, V., Muller, E.: Timing, diffusion, and substitution of successive generations of technological innovations: The IBM mainframe case. Technological Forecasting and Social Change 51, 109–132 (1996)
24. Shen, W., Altinkemer, K.: A multigeneration diffusion model for IT-intensive game consoles. Journal of the Association for Information Systems 9, 20 (2008)

25. Van Heerde, H.J., Gupta, S.: The Origin of Demand: A System to Classify the Sources of Own-Demand Effects of Marketing Instruments (2005)
26. Deleersnyder, B., Geyskens, I., Gielens, K., Dekimpe, M.G.: How cannibalistic is the Internet channel? A study of the newspaper industry in the United Kingdom and the Netherlands. International Journal of Research in Marketing 19, 337–348 (2002)
27. Lomax, W., McWilliam, G.: Consumer response to line extensions: Trial and cannibalisation effects. Journal of Marketing Management 17, 391–406 (2001)
28. Mahajan, V., Sharma, S., Buzzell, R.D.: Assessing the impact of competitive entry on market expansion and incumbent sales. The Journal of Marketing 57, 39–52 (1993)
29. Kaiser, U.: Magazines and their companion websites: competing outlet channels? Review of Marketing Science 4, 1–24 (2006)
30. Simon, D., Kadiyali, V.: The effect of a magazine's free digital content on its print circulation: Cannibalization or complementarity? Information Economics and Policy 19, 344–361 (2007)
31. Ghose, A., Smith, M.D., Telang, R.: Internet Exchanges for Used Books: An Empirical Analysis of Product Cannibalization and Welfare Impact. Information Systems Research 17, 3–19 (2006)
32. Danaher, B., Dhanasobhon, S., Smith, M.D., Telang, R.: Converting Pirates Without Cannibalizing Purchasers: The Impact of Digital Distribution on Physical Sales and Internet Piracy. Marketing Science 29, 1138–1151 (2010)
33. Smith, M.D., Telang, R.: Piracy or promotion? The impact of broadband Internet penetration on DVD sales. Information Economics and Policy 22, 289–298 (2010)
34. Hashim, M.J., Tang, Z.: Delivery of Movies via the Internet: Uncovering Strategies for the Release of Digital Formats (2010)
35. Meredith, L., Maki, D.: Product cannibalization and the role of prices. Applied Economics 33, 1785–1793 (2001)
36. Yuan, Y., Capps, O.J., Nayga, R.M.J.: Assessing the Demand for a Functional Food Product: Is There Cannibalization in the Orange Juice Category? Agricultural and Resource Economics Review 38, 153–165 (2009)
37. Albuquerque, P., Bronnenberg, B.J.: Estimating Demand Heterogeneity Using Aggregated Data: An Application to the Frozen Pizza Category. Marketing Science 28, 356–372 (2009)
38. Page, A.L., Rosenbaum, H.F.: Redesigning Product Lines with Conjoint Analysis: How Sunbeam Does It. Journal of Product Innovation Management 4, 120–137 (1987)
39. Kannan, P.K., Pope, B.K., Jain, S.: Pricing Digital Content Product Lines: A Model and Application for the National Academies Press. Marketing Science 28, 620–636 (2009)
40. Stair, R.M., Reynolds, G., Reynolds, G.W.: Fundamentals of Information Systems. Cengage Learning (2008)
41. Hoch, D.J., Roeding, C.R., Purkert, G.: Secrets of software success: Management insights from 100 software firms around the world. Harvard Business Press, Boston (2000)
42. Filistrucchi, L.: The impact of Internet on the market for daily newspapers in Italy (2005)
43. Smith, M.D., Telang, R.: Competing with free: The impact of movie broadcasts on DVD sales and Internet piracy. MIS Quarterly 33, 321–338 (2009)
44. Novelli, F.: Platform Substitution and Cannibalization: The Case of Portable Navigation Devices. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 141–153. Springer, Heidelberg (2012)
45. Novelli, F., Wenzel, S.: Adoption of an online sales channel and appification in the enterprise application software market: A qualitative study. In: Proceedings of the 21st European Conference on Information Systems, Utrecht (2013)

Determinants and Dynamics of Technology-Related Acquisitions: The Case of Software-Based Industries

Marcus Wagner

Chair for Entrepreneurship and Management, Julius Maximilians University
Stephanstr. 1, 97070 Wuerzburg, Germany
marcus.wagner@uni-wuerzburg.de

Abstract. In high technology industries the option to use acquisitions as a means for technology sourcing is important. This paper investigates the determinants and dynamics of this for the context of software-based high technology industries specifically as concerns the association of acquirer characteristics with acquisition behavior and finds a substitutive relationship between acquisitions and own research activities and between acquirer and target patenting. Consistent with this firms with low patenting intensity acquire targets with many prior patents, which indicates that aiming to increase the overall efficiency of an incumbent's R&D process by acquiring were she has the biggest weaknesses is most successful. For a subset of acquisitions and acquirers, these findings are related in more detail to target characteristics.

1 Introduction and Research Questions

Innovation in large firms may be hindered by organizational inertia or lacking knowledge [1]. It seems that the acquisition of technology-rich targets can address this. The feasibility of this approach seems to depend however on the nature of the industry considered. One important characteristic of software-based industries is their low capital intensity, which means that startups do not require large investments in fixed assets to enter. As a result of this, the total asset value of the average entrant in software-based industries is relatively low and therefore a complete acquisition of a technology-rich target is more easily feasible and hence more likely (compared to just a minority shareholding in such a target). This paper analyses the acquisition dynamics that derive from this based on econometric and descriptive analyses.

The research questions underlying the analysis derive from a literature review and focus on the factors motivating larger incumbents to acquire smaller firms and startups in software-based industries. The relevance of acquisitions for substituting research and development (R&D) weaknesses can be linked to theoretical arguments concerning obstacles to innovation in larger firms, empirical research providing examples why firms may not be able or willing to carry out specific types of innovation [2]. Obstacles to innovation can emerge in the sense that larger firms are not able to carry out specific innovations, as is also shown in theoretical models [3]. One response of firms to not being able to carry out an innovation at acceptable cost can be the acquisition of technology-rich targets in order to make up for missing

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 43–47, 2013.

capabilities. The paper extends the knowledge about this by clarifying which factors determine the acquisition of younger, technology-rich target firms and how the pattern of acquisitions evolves (i.e., what acquisition dynamics exist and how they can be explained). Based on these considerations, two sets of research questions emerge:

1. Is acquisition in software-based industries motivated by substitutiveness or complementarity with own R&D? Does the association differ depending on the dependent variable (number of acquisitions versus technological value of targets)?
2. Do acquisitions relate mainly to specific technology segments and when are targets acquired? Can timing and segment specificity be related to the substitutive versus complementary nature of acquisitions?

2 Methodology

The empirical analysis focusses on a specific software-based, namely Electronic Design Automation (EDA), a sub-segment of the global semiconductor industry that has very beneficial characteristics for addressing the above research questions since it is characterized by continued acquisition and innovation. Given the exploratory nature of this study and the above research questions the analysis initially employs statistical and econometric analysis to address the more aggregate quantitative aspects of the above research questions based on primary data collected from established sources such as the SDC Platinum and Worldscope Disclosure databases and the U.S. Patent and Trademark Office. This data covers the acquisitions by the 14 largest firms in the EDA industry during the period of 1981 until 2005 and firm characteristics. To enable an analysis of the remaining aspects of above research questions, the aggregate quantitative analysis is supplemented on the second stage by a more qualitative analysis of a subset of firms for which was augmented with secondary data from trade journals, industry publications, company websites and a content analysis of Securities and Exchange Commission filings.

3 Results

In order to address the first set of research questions posed, regression models were estimated for extent (total number of acquisitions, see Table 1) and degree of technological knowledge acquisition (as measured by the total number of patents granted to the acquired firms in the acquisition year and the five years prior to it, see Table 2). Since both dependent variables are count data and the data is of panel nature a negative binomial random-effects (RE) model is used based on the Hausman test.

Overall, the results show that patenting intensity has consistently a significant negative association with both dependent variables. This supports the notion that acquisitions compensate weak innovation output indicated by lower levels of acquirer patenting intensity. Notably, the coefficient of the patenting intensity in Table 2 is considerably greater as in Table 1. Furthermore, the argument, that acquisition of innovation is a substitute for own R&D efforts is supported as concerns the number of acquisitions.

Table 1. Negative binomial model, dependent variable: total number of acquisitions

Variables	RE estimates
Financial leverage (total assets to total equity)	-0.4462 (0.3216)
Current ratio (current assets to current liabilities)	-0.0772 (0.1194)
Sales growth (% over previous year)	0.0022 (0.0034)
Sales (natural logarithm of net sales in mn €)	0.4045 (0.1296)***
R&D intensity (R&D expenditure to net sales in %)	-0.0463 (0.0218)**
Missing R&D intensity data (dummy; 1 = missing)	-29.4861 (1218611.5)
Patenting intensity (Patents granted to net sales in %)	-0.0233 (0.0127)*
Missing patenting intensity data (dummy; 1 = missing)	-0.2490 (0.5038)
Company headquartered in Europe (dummy; 1 = yes)	-0.1852 (0.5309)
Company headquartered in Asia (dummy; 1 = yes)	-1.0021 (0.6245)
Constant	14.7703 (637.8766)
Log-likelihood	-114.63298
No. of observations (firms)	105 (14)
Wald Chi²	42.86***
Hausman specification test Chi²	<0.01

Notes: Significance levels: * p < 0.1; ** p < 0.05; *** p < 0.01; Country base category: U.S.; Likelihood-ratio test vs. pooled: Chi² = 0.22, p-value >= Chi² = 0.318

Table 2. Negative binomial model, dependent variable: patents granted to targets

Variables	Random effects estimates
Financial leverage (total assets to total equity)	0.2068 (0.4146)
Current ratio (current assets to current liabilities)	-0.3507 (0.3348)
Sales growth (% over previous year)	-0.0083 (0.0102)
Sales (natural logarithm of net sales in mn €)	1.0966 (0.2969)***
R&D intensity (R&D expenditure to net sales in %)	0.0097 (0.0142)
Missing R&D intensity data (dummy; 1 = missing)	-24.3519 (203473.2)
Patenting intensity (Patents granted to net sales in %)	-0.0663 (0.0261)**
Missing Patenting intensity data (dummy; 1 =missing)	-0.5224 (1.0713)
Company headquartered in Europe (dummy; 1 = yes)	2.2475 (0.9131)**
Company headquartered in Asia (dummy; 1 = yes)	0.7629 (1.3680)
Constant	-8.3025 (2.4877)***
Log-likelihood	-122.23788
No. of observations (firms)	105 (14)
Wald Chi²	22.56**
Hausman specification test Chi²	<0.01

Notes: Significance levels: * p < 0.1; ** p < 0.05; *** p < 0.01; Country base category: U.S. ; Likelihood-ratio test vs. pooled: Chi² = 0.00, p-value >= Chi² = 1.000

In order to address the second set of the above research questions which refer to the type of target and other qualitative aspects the following Table 3 summarises for the three largest firms in the industry if acquired targets are in similar fields of technology. Based on detailed descriptions and target and acquirer Standard Industry

Classification (SIC) codes, the technology segment most relevant for each individual acquisition was identified. The percentage figures in Table 3 give the share in the total of the acquisitions of the acquiring firm denoted in the top that was related to technology segments were also the other firms made acquisitions (i.e. homogeneity). The right column of Table 3 provides the number of acquisitions, per year, that took place in the same technology segment out of total acquisitions of three largest firms.

Table 3. Homogeneity of acquisitions of the top 3 EDA firms based on technology segments

Year	Cadence	Mentor	Synopsys	% acquisitions in same segments out of total
1989	0%	0%	0%	0%
1990	100%	100%	0%	67%
1991	0%	0%	0%	0%
1992	0%	0%	0%	0%
1993	0%	0%	0%	0%
1994	100%	100%	0%	67%
1995	0%	0%	0%	0%
1996	100%	63%	100%	70%
1997	67%	0%	67%	67%
1998	14%	100%	33%	36%
1999	0%	0%	0%	0%
2000	0%	0%	0%	0%
2001	0%	0%	0%	0%
2002	33%	25%	33%	30%
2003	66%	20%	0%	27%
2004	40%	50%	0%	27%
2005	0%	0%	0%	0%

As can be seen, the overlap in technology segments is very limited. Only in 8 of the 17 years analysed, an overlap was found at all. Even on these occasions, the overlap never constituted more than 70% of all acquisitions made by the three largest EDA firm in that year and in 4 of these 8 years was below 40%. This indicates that acquiring technology that is novel to the industry as a whole is not dominating the observed acquisitions, even though it may be a partial explanation in some years (especially 1990, 1994, 1996 and 1997). This suggests that several reasons matter simultaneously, which also the total number of acquisitions per segment indicates. In all segments except one at least two firms did acquisitions and in one third of the segments all three large firms acquired. Furthermore, the number of acquisitions per segment differs only up to 60% between firms which was however only the case in one segment (in four segments the difference was 0%, in three it ranged between 28% and 50%, and in one only one firm acquired). All of the three largest firms acquired from 1989 to 2005 in three segments (in which, respectively, 11%, 50% and 14% of all acquisitions during this period took place) and only in one segment, only one of the three largest firms acquired and the other two never did an acquisition (representing 2% of all acquisitions of the largest three EDA firms from 1989 to 2005). There is lesser evidence of temporal clustering of acquisitions in specific technologies. In some segments

acquisitions concentrate on shorter time periods (in all segments acquisitions take place in at least two years, in two in six years, in one each in 10 and 15 years, respectively). In four out of nine segments, the time between the first and last acquisition is more than half of the 17 years studied. In four years (1989, 1990, 2000, 2005), all acquisition targets were from only one segment, to which also the majority of targets across all years belonged. In all other years, acquisitions took place in at least two (1991, 1992, 1994, 1995, 2001), three (1993, 1997), four (2002-2004), five (1999), six (1996) and seven (1998) segments. As concerns timing of acquisitions, based on 80 acquisitions (i.e. those of the largest three firms during 1989 to 2005 for which age data was available), a skewed distribution of target age at the time of acquisition is found: the mean age of targets at the time of acquisition is 7.62, the median is 5 years (standard deviation: 5.42 years).

4 Conclusions and Discussion

This paper contributes to a better understanding of the acquisition dynamics in software-based industries with a focus on the relevance of acquirer R&D and homogeneity and timing of acquisitions. As concerns the latter two, the analysis of data for the largest three firms in the EDA industry revealed no high homogeneity of acquisitions: each large incumbent acquires not selectively, but over a broader range of segments. This is consistent with the substitutiveness of acquisitions with regard to own R&D efforts and outputs found in the aggregate quantitative analysis and suggested by qualitative mechanisms that were developed in prior work [1]. Since the efforts and outputs are generic, acquisitions should also be generic, i.e. across segments, which the analysis of the enriched data for the largest three firms confirms. Compared to this, the issue of timing is less clearly resolved in that both, acquisitions early on or later when a target is proven in the market seem feasible and are also identified in the data analysed. This is also consistent with the aggregate quantiative analysis, in that generic R&D efforts and outputs, allow differing timing and temporal dispersion. From a practitioner view, the results of the analysis suggest that acquirers should use acquisitions to compensate their main weaknesses. More specifically, results suggest that firms differentiate between input and output aspects of R&D performance. Firms with low patenting intensity acquire targets with many prior patents, which is consistent with the significantly stronger negative coefficient of patenting intensity and an insignificant coefficient for R&D intensity when prior target patenting is the dependent variable and managers can use this as guidance.

References

1. Henderson, R., Clark, K.B.: Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. Admin. Sci. Q. 35, 9–30 (1990)
2. Hauschildt, J.: Opposition to innovations – destructive or constructive? In: Brockhoff, K., Chakrabarti, A., Hauschildt, J. (eds.) The Dynamics of Innovation, pp. 217–240. Springer, Berlin (1999)
3. Henkel, J., Rønde, T., Wagner, M.: And the Winner is – Acquired. Entrepreneurship as a Contest with Acquisition as the Prize. Discussion Paper No. DP8147, Center for Economic Policy Research (CEPR), London (2012)

Engineering Open Innovation – Towards a Framework for Fostering Open Innovation

Krzysztof Wnuk and Per Runeson

Department of Computer Science, Lund University, Sweden
{Krzysztof.Wnuk,Per.Runeson}@cs.lth.se
http://serg.cs.lth.se

Abstract. Open innovation is an emerging innovation paradigm that can greatly accelerate technical knowledge innovation in software companies. The increasing importance and density of software in today's products and services puts extensive pressure on excelling the discovery, description and execution of innovation. Despite that, software engineering literature lacks methods, tools and frameworks for full exploitation of technological advantages that open innovation can bring. This paper proposes a software engineering framework, designed to foster open innovation by designing and tailoring appropriate software engineering methods and tools. Furthermore, this paper discusses the methodological and process dimensions and outlines challenge areas that should be reviewed when transitioning to software engineering driven open innovation.

Keywords: open innovation, software engineering framework, literature study, methodological and process study.

1 Introduction

The development of software products is mainly driven by *innovation* [1]; i.e. the novel utilization of technical knowledge to develop new products and services. For example, Volvo, the truck company, estimates that 90% of new innovations are in the field of electronics, and 80% thereof is software[1]. Similarly, most of the innovation and resulting market success at Siemens originates from software [2].

A majority of the innovation within software intensive products is implemented in software and increasingly dependent on a new paradigm called *Open Innovation*(OI), which typically, but not necessarily is implemented using *Open Source Software* (OSS). In recent years, the influence of OI has become significant in the development and evolution of software products and services, e.g. in the Android ecosystem. OI implies that no single firm or other actor is sufficient for developing new products and services; instead several loosely connected organizational actors interplay. The OI context is characterized by: (1) collaborative efforts over single company/person work, (2) loose connections over contractual

[1] http://www.swedsoft.se/Swedsoft_SRA_2010.pdf

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 48–59, 2013.

agreements, (3) demonstrated results over predictions and (4) bottom-up specification approaches. These differences in characteristics make software engineering practices significantly challenging.

Software, with its flexibility in multiple aspects, is an excellent enabler for innovation. On the other hand, this flexibility must be managed, not to lose control over the software. Hence, software engineering (SE) in an OI context is a major research challenge since engineering practices for in-house, contract-based development may not be feasible. Therefore, we set out to define a framework to support software engineering for open innovation.

This paper presents result from an exploratory literature study that constitutes the first step of our efforts towards building a framework that *aims to synthesize a scientifically founded software engineering framework for open innovation*. We review existing literature and map existing research as a basis for new research [3]. Our goal is to develop new or adapt existing practices into a framework that meet challenges in the multi-organizational, heterogeneous open innovation context.

This paper is structured as follows: Section 2 outlines definitions and background, Section 3 presents the literature review results. Section 4 outlines the engineering open innovation framework while Section 5 outlines future research directions and concludes the paper.

2 Background and Definitions

Open innovation was introduced by Chesbrough [4] as "a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology" [4]. A more recent definition of open innovation by Lichtenthaler [5] focuses on "systematically relying on a firm's dynamic capabilities of internally and externally carrying out the major technology exploitation and acquisition tasks along the innovation process". This increased external stream of knowledge may result in more *disruptive* innovations (i.e. taking large steps towards something new) coexisting with *sustaining* innovations (i.e. continuously improving solutions to create more value [6,7]). This, in turn, increases the pressure for software engineering methods that can cope with increased interoperability, flexibility and significantly enhanced engineering characteristics.

The OECD defines four main types of innovation in the Oslo manual [8], *product, process, marketing* and *organizational*. The inherent characteristics of software enable novel approaches to all four types of innovation. Product innovation (the software itself) may bring new value to customers at negligible production and distribution costs. Process innovation involves new means of developing software, e.g. OSS communities. Marketing innovations include new business models, e.g. offering services at the price of being exposed to ads or sharing information. Organizational innovation includes new ways to work across different actors, where open innovation is an example. All four types are interconnected and therefore have to be researched in context.

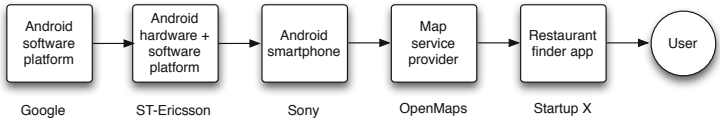

Fig. 1. The Android eco-system as an example of a (partly) open innovation value chain

An example of a (partly) open innovation value chain is illustrated through the Android eco-system in Figure 1. Google provides the (semi-)open Android software platform to device suppliers, which in turn adapt the platform to their and their customers' specific needs. The Android platform contains million lines of code adapted for hundreds of products, various hardware products, wireless network standards and database systems. Service providers may offer map services tailored to the Android phones (e.g. OpenMaps), which other app providers may utilize to derive specialized map services, e.g. a restaurant finder by a local startup company. This innovation chain is driven by several actors in collaboration and dependencies on one another, with different individual goals, time scales, size of organization, techniques etc. As one actor in the chain evolves its parts, others must follow, but no single actor is in full control.

OSS is a mechanism that may embody the principles of open innovation. OSS is not a new phenomenon on its own, however the novelty in recent years is the widespread adoption in industry, where closed innovation used to be the dominating paradigm, which requires new approaches to software development [9]. A recent literature survey about OSS identified a research gap in the area of OSS and open innovation [10]. Software ecosystems could also be one of the types of open innovation where a network of collaborators constituting an ecosystem is open [11]. Finally, there is a need for processes supporting large-scale development with open innovation; companies typically apply the same processes as used in closed innovation [9], while there certainly are opportunities for more tailored processes to improve the efficiency and effectiveness of the development [12].

3 Literature Review

We conducted a literature review using a hybrid approach by combining database search (Compendex and Inspect), and snowball sampling. We selected a mapping study approach because our main goal was to explore the area rather than synthesize the current state of the art [3]. We used the following search queries (searched in titles, abstracts or subjects):

- open innovation AND requirements engineering
- open innovation AND software design
- open innovation AND software development
- open innovation AND software testing
- open innovation AND software
- software engineering AND innovation
- methodology OR method AND open innovation AND software

Table 1. Classification of papers; research type according to Wieringa et al. [14]

Research type	Techniques	Tech & Proc	Processes	Methods
Validation	[15]	[16]		
Evaluation[1]		[17]	[18]	[19]
Solution	[20]	[21] [22]	[23] [24] [25] [26] [27] [28] [29] [30] [31] [11]	
Conceptual[2]	[32]		[33]	
Opinion	[34] [35]	[36] [37]		[38] [39]
Experience		[40] [41] [42]	[43] [44] [45]	

[1] The original classification [14] only covers engineering research, while we here classify also observational empirical studies as 'Evaluation', as they evaluate current practice.
[2] Called 'Philosophical' originally.

- open innovation AND software as a service OR saas
- open innovation AND software AND eco system
- innovation AND software AND eco system

The above queries returned 1480 records that we checked by reading titles and, if in scope, also abstracts. 32 papers were selected for full reading. We categorized these papers into two dimensions; the first dimension categorized the articles according to the topic of *techniques, processes* and *research methods* while the second dimension of research type was created based on systematic mapping guidelines [13,14]. The classification is summarized in Table 1.

3.1 Software Engineering Techniques for Open Innovation

Five papers discussed or suggested a specific software engineering method or technique for an open innovation context. El-Sharjawy and Schmid proposed and experimentally evaluated an approach for deriving creative triggers from a knowledge map of requirements [15], in a paper of validation type. We identified two opinion papers: Petrenko and Petrenko discussed the challenges of using formal methods to analyze requirements and work with legacy code that can foster what they called Innovation Economy [34] while Grube and Schmid suggested which creativity techniques are appropriate for requirements engineering [35]. A conceptual paper by Kauppinen et al. argued that practitioners do not see requirements engineering as a creative process and suggested focusing on "unarticulated needs" to unlock more innovation from requirements engineering processes and techniques [32]. A solution using social networks to document ideas and thus foster open innovation was proposed by Singer et al. [20].

The remaining 9 papers in the Techniques category were also touching upon the Processes category (see Section 3.2). We found two opinion papers: one focusing on how to avoid innovation lock-in from a pre-planned variability model of a software product line [36] and one focusing on sharing the source code and opening bug-tracking tools with the clients [37]. Next, an experience paper by Copeland suggested new ways of communicating the information about testing [40]. Theodore et al. studied how outsourcing can inject tangible forms of

innovation [41] presenting an example of innovative testing methods (unfortunately without detail about the methods) that originated from such a collaboration and improved time-to-test by 90% and reduced cost by 70%.

Five papers, that we categorized in both the Techniques and Processes categories, focused on software development. One experience paper focused on investigating how software startups can use opportunistic and pragmatic reuse to develop innovative products [42]. One evaluation paper focused on how agile development processes can become more open by utilizing outside-in and inside-out process [17] models. Two solution papers proposed giving developers more authority on *when* and *how* to use innovative software development techniques [21] and utilizing prototyping, agile methods, developers using products they develop and sharing knowledge to foster innovation [22]. Finally, one validation paper experimentally concluded that leaving the developers free to define their own development processes is beneficial from the innovation diffusion perspective [16].

3.2 Software Engineering Processes That Foster Open Innovation

We categorized 14 articles as only concerning the software engineering processes category. Ten papers presented various solutions, among which three supported innovation selection processes by an audition-inspired process for screening, refining and selecting the most promising innovations [23], a knowledge management scheme that supports transition of software innovations (also legacy systems) to enterprise systems [24] and a method based on neuro-fuzzy decision trees for innovation projects selection [25]. The prototyping approach to open innovation was explored in two publications: Eklund and Bosch suggested turning the entire R&D process into an innovation experiment system with direct customer involvement in design decisions [26] while Bullinger et al. proposed an open prototyping solution [27]. Misra et al. advocated using a GQM-based method to derive a measurement framework for software innovation process [28] while Felfernig et al. proposed utilizing artificial intelligence for open innovation in e-government contexts [29]. Jansen [11] focused on measuring the degree of openness of a software organization. Two publications proposed solutions to explore open innovation communities by visualizing people-innovation-networks [30] or modeling service systems in terms of communities of co-innovation [31].

Among experience papers in this category, Hanssen [43] reported lessons learned from opening up a software product line, observing that it improved the ability to catch tacit requirements (related to unarticulated needs mentioned by Kauppinen et al. [32]. Yilmaz discovered, based on a simulation, that decentralized coordination schemes as well as moderate degrees of assertiveness result in a higher incidence of innovation for open source software communities [44]. Carrero [45] discussed how service delivery platforms enable service providers to achieve open innovation.

In a conceptual paper, Lyytinen and Damsgaard observed that the six conjectures of diffusion of innovation need to be revisited for complex and networked IT systems and additional issues should be considered [33]. In an evaluation paper, Lane et al. concluded that finding a good balance between art and science,

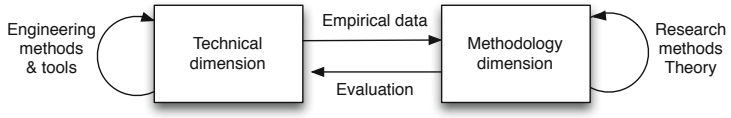

Fig. 2. Overview of framework dimensions and their relations

allowing failures as a part of learning process, and trying different approaches are important success factors for software innovative process [18]. To summarize, although ten papers presented solutions [11,23,24,25,26,27,28,29,30,31] only one evaluation paper was identified [18].

3.3 Software Engineering Research Methodologies

An opinion paper by Bayer and Melone discussed challenges in applying diffusion theory for software engineering technological innovations, outlining seven limitations [38]. Prechelt and Oezbek outlined a research method solution for studying open source software process innovation, suggesting that grounded theory is feasible for deriving mini-theories about process innovation, but not focusing strictly on open innovation [39]. In an evaluation paper, Rossi et al. proposed a quantitative instrument (based on stochastic models) for measuring the assimilation gaps in IT innovation [19]. No solution paper was identified in this category.

4 Engineering the OI–SE Framework

The literature review in Section 3 brings supporting evidence for our research efforts. We identified only three evaluation papers, see Table 1, none strictly focusing on open innovation and mostly reporting exploratory evaluations [17,18,19]. Both identified validation papers focused on internally derived innovations [15,16]. Among 12 identified solution papers, only four are devised for open innovation [29,30,31,11].

We address the open innovation issues from a product and process innovation point of view, in the intersection with marketing and organizational innovation perspectives [8]. Based on the literature review, we propose a framework having two main dimensions, one *technical* and one *methodological*. The dimensions are mutually dependent, as the technical dimension is the empirical basis for the methodological part, and the methodological part is needed for the technical part. Figure 2 gives an overview of the project and its dimensions, which is inspired by Hevner's design science model [46] and Wieringa et al [47].

4.1 Technical Dimension

The technical dimension has two interrelated parts, see Figure 2: 1) software engineering techniques, such as requirements engineering, software design, software development and software testing techniques, and 2) software engineering

processes as an integrating part of the above mentioned techniques. We believe (providing empirical evidence for our hypotheses is one of the goals of the framework) that using appropriate techniques and analyzing the outcomes of these techniques can foster open innovation.

Requirements: requirements engineering research has evolved from concentrating on the specification problem in the 1990's to pursuing wide and open-ended investigations of the conception and strategic evolution of software in relation to decision-making on enterprise, product/service and project levels. The integrated strategic and tactical decision-making needed in large-scale engineering projects is a key challenge for software engineering for open innovation [48]. However, papers identified during the literature review focus mainly on supporting the discovery of innovation [15,20,32,34].

We build on previous research and focus on release-planning [48,49], stakeholder analysis, trade-off between effort (cost) and value and the degree of innovation in candidate features needed in evolving systems, to take significant future market shares in open innovation software development [48,50].

Design: efficient software architectures for open innovation should enable open and seamless integration of externally acquired modules. Among idenfitied papers, Böckle [36] postulated that software product lines are generally hindering innovation and that variability locks in innovation as it focuses on reusing the same code and thus minimizing creative adaptations. Moreover, software product lines are designed with a premise that the same code will be used for a long time, which directly hinders disruptive innovation. Similarly, a software design with high coupling may be hindering open innovation as new modules and sub-systems could not be easily integrated. Thus, there seems to be a need for evolution from traditional SPLs toward software ecosystems [36] which introduced necessary flexibility in an organizational matter.

Development: is pair programming going to result in more innovation that other programming techniques? This is just an example question that should be investigated in the framework. Green suggested [21] that giving developers more authority on when to use the development technique innovations helps to actually use them rather than drop them. Sharing the source code and bug-tracking system [37] or open prototyping [28] also seem to be fostering innovation. However, identified studies focus on development processes rather than techniques [16,17,21,28,22,37,26]. Among identified studies, Jansen [11] presented a model for establishing the degree of openness of a software organization. Further empirical investigations are needed to yield concrete examples of which development techniques foster open innovation.

Testing: Software testing in open innovation has a dual role: 1) to verify functions and characteristics of open components and services, supplied by others, and 2) to verify functions and characteristics of services delivered to stakeholders higher up in the value chain; ultimately end users. Since specifications and contracts are sparse in the open innovation context, they have to be defined otherwise.

Test driven development is a method which has proven feasible in a dynamic practice [51]. Its combination of specification and test [52] can be tailored for use in open innovation. Software engineering in open innovation tends to be very iterative, and thus *regression testing* is a key issue. We build on previous findings [53,54] and adapt test selection and prioritization approaches to open innovation, as well as using it to detect changes in the environment [55].

Efficient Processes: The move towards agile processes have substantially changed software engineering practices during the last decade. The efficiency of some practices have been empirically demonstrated [56], and the positive attitudes of engineers "being in control" are witnessed [21,22,55]. Still, the efficiency of agile (or other) practices in open innovation is not targeted; our review identified only one evaluation paper [17] and one validation paper [16] in this area. Further, most studies either focus on the small *or* the large context, e.g. [23,24,26,43], but in *OI* we have small contexts within the larger context.

We plan to investigate the issue of stakeholders and stakeholder representatives in an open innovation context [55]. Further, mechanisms for synchronizing different actors in the open innovation value chain are planned to be researched [48], based on observations of different actors in open source projects [57]. We also plan to explore how open source practices can support openness across groups and other organizational borders within closed organizations. Findings should be embodied in practice models for small actors in a large context, enabling growth, as previously done on testing practices for small companies [58].

4.2 Methodological Dimension

Studying software engineering for open innovation must be an empirical endeavor since we are addressing complex phenomena in the real world. The question is which type of empirical study is to be conducted.

Prechelt and Oezbek conducted four studies on OSS process innovation, and concluded that using mailing list archives was the most efficient research method, compared to direct participation and polling developers for data [39]. The easy access to electronically searchable information is probably one of the key reasons for the popularity on conducting research in OSS archives, independently of whether purpose of the research is open or proprietary software [59].

The scope of the open innovation framework is wider than OSS projects only. This, combined with general advice on using research method triangulation, lead us to propose combining *archival studies* with *participant-observer* studies [60], to enable insights into the dynamics of the open innovation.

Building synthesized knowledge from the empirical observation needs replication of studies [61], synthesis of findings from several empirical studies [62], as well as work in theory building [63]. However, these needs are general for software engineering, and not specific to the open innovation aspects.

5 Conclusions

Our review of the related literature remains incomplete (the results presented in Section 3 are preliminary). However, we can identify three research areas where both researchers and practitioners can benefit: (1) providing practical guidelines for selecting the most appropriate requirements engineering, software design, software development and software testing techniques for open innovation, (2) researching software engineering processes that support open innovation and (3) finding new research methodologies for conducting software engineering research in open innovation contexts.

In the intersection between different types of innovation (product, process, marketing and organizational [8]), there is a significant potential for software innovations. Especially the intersection between software engineering and open innovation lacks empirical research, as shown in our literature review, which we hope and aim that our research framework will foster.

Future work should focus on investigating if the presented framework could support the intrinsic creativity and unpredictability of innovation. Furthermore, we plan to explore and possible identify activities that can not be clearly categorized into the four traditional software development process steps. As we only searched Compendex and Inspect, more databases should be searched and the literature review should be replicated in a systematic way. Finally, we plan to investigate the possible relationships between the business models and open innovation.

Acknowledgements. The authors would like to thank David Callele for excellent reviews of this paper. This work is founded by the SYNERGIES project, Swedish National Science Foundation, grant 621-2012-5354.

References

1. Quinn, J.B., Baruch, J.J., Zien, K.A.: Software-based innovation. Sloan Management Review 37(4), 11–24 (1996)
2. Achatz, R.: Product line engineering at siemens – challenges and success factors: A report on industrial experiences in product line engineering. In: 2011 15th International Software Product Line Conference (SPLC), pp. 10–11 (August 2011)
3. Kitchenham, B.A., Budgen, D., Brereton, O.P.: Using mapping studies as the basis for further research - A participant-observer case study. Information & Software Technology 53(6), 638–651 (2011)
4. Chesbrough, H.: Open Innovation: The new imperative for creating and profiting from technology. Harvard Business School Press, Boston (2003)
5. Lichtenthaler, U.: Open innovation in practice: An analysis of strategic approaches to technology transactions. IEEE Trans. on Eng. Mgmt 55(1), 148–157 (2008)
6. Khurum, M., Gorschek, T., Wilson, M.: The software value map — an exhaustive collection of value aspects for the development of software intensive products. Journal of Software: Evolution and Process, n/a–n/a (2012)
7. Khurum, M., Aslam, K., Gorschek, T.: A method for early requirements triage and selection utilizing product strategies. In: Proc. of the 4th Asia-Pacific Software Engineering Conf. APSEC 2007, pp. 97–104. IEEE Computer Society (2007)

8. Oslo Manual – Guidelines for collecting and interpreting innovation data. 3rd edn. OECD and Eurostat (2005)
9. Höst, M., Oručević-Alagić, A., Runeson, P.: Usage of open source in commercial software product development - findings from a focus group meeting. In: Caivano, D., Oivo, M., Baldassarre, M.T., Visaggio, G. (eds.) PROFES 2011. LNCS, vol. 6759, pp. 143–155. Springer, Heidelberg (2011)
10. Höst, M., Orucevic-Alagic, A.: A systematic review of research on open source software in commercial software product development. Information and Software Technology 53(6), 616–624 (2011)
11. 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)
12. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and mozilla. ACM Trans. Softw. Eng. Methodol. 11(3), 309–346 (2002)
13. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proc. 12th Int. Conf. Evaluation and Assessment in Soft. Eng. EASE 2008, pp. 68–77. British Computer Society, UK (2008)
14. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirement Engineering 11, 102–107 (2006)
15. El-Sharkawy, S., Schmid, K.: A heuristic approach for supporting product innovation in requirements engineering: a controlled experiment. In: Berry, D. (ed.) REFSQ 2011. LNCS, vol. 6606, pp. 78–93. Springer, Heidelberg (2011)
16. Tortorella, M., Visaggio, G.: Empirical investigation of innovation diffusion in a software process. International Journal of Software Engineering and Knowledge Engineering 9(5), 595–621 (1999)
17. Conboy, K., Morgan, L.: Beyond the customer: Opening the agile systems development process. Inf. and Soft. Techn. 53(5), 535–542 (2011)
18. Lane, J.A., Boehm, B., Bolas, M., Madni, A., Turner, R.: Critical success factors for rapid, innovative solutions. In: Münch, J., Yang, Y., Schäfer, W. (eds.) ICSP 2010. LNCS, vol. 6195, pp. 52–61. Springer, Heidelberg (2010)
19. Rossi, B., Russo, B., Succi, G.: Path dependent stochastic models to detect planned and actual technology use: A case study of openoffice. Inf. and Soft. Techn. 53(11), 1209–1226 (2011)
20. Singer, L., Seyff, N., Fricker, S.A.: Online social networks as a catalyst for software and it innovation. In: Proc. of the 4th Int. Workshop on Social Soft. Eng. SSE 2011, pp. 1–5. ACM, New York (2011)
21. Green, G., Hevner, A.: The successful diffusion of innovations: guidance for software development organizations. IEEE Soft. 17(6), 96–103 (2000)
22. Moe, N.B., Barney, S., Aurum, A., Khurum, M., Wohlin, C., Barney, H.T., Gorschek, T., Winata, M.: Fostering and sustaining innovation in a fast growing agile company. In: Dieste, O., Jedlitschka, A., Juristo, N. (eds.) PROFES 2012. LNCS, vol. 7343, pp. 160–174. Springer, Heidelberg (2012)
23. Gorschek, T., Fricker, S., Palm, K.: A lightweight innovation process for software-intensive product development. IEEE Soft. 27(1), 37–45 (2010)
24. Corbin, R.D., Dunbar, C.B., Zhu, Q.: A three-tier knowledge management scheme for software engineering support and innovation. Journal of Systems and Software 80(9), 1494–1505 (2007)

25. Hongxia, J., Jianna, Z., Xiaoxuan, C.: The application of neuro-fuzzy decision tree in optimal selection of technological innovation projects. In: Eighth ACIS International SNPD Conference, vol. 3, pp. 438–443 (August 2007)
26. Eklund, U., Bosch, J.: Architecture for large-scale innovation experiment systems. In: Working IEEE Conf. on Soft. Architecture, pp. 244–248 (2012)
27. Bullinger, A.C., Hoffmann, H., Leimeister, J.M.: The next step - open prototyping, Helsinki, Finland (2011)
28. Misra, S.C., Kumar, V., Kumar, U.: Goal-driven measurement framework for software innovation processes. In: Arabnia, H.R., Reza, H. (eds.) Proc. of the Int. Conf. on Soft. Eng. Research and Practice, pp. 710–716. CSREA Press (2005)
29. Felfernig, A., Russ, C., Wundara, M.: Toolkits supporting open innovation in e-government. In: Proc. of the Sixth Int. Conf. on Enterprise Information Systems, Porto, Portugal, pp. 296–302 (2004)
30. Friess, M., Groh, G., Reinhardt, M.: Supporting open innovation communities by an interactive network visualization. In: Proceedings of the IADIS International Conferences, New York, NY, USA, pp. 23–28 (2010)
31. Janner, T., Schroth, C., Schmid, B.: Modelling service systems for collaborative innovation in the enterprise software industry - the st. gallen media reference model applied. In: IEEE Int. Conf. on Services Computing, vol. 2, pp. 145–152 (2008)
32. Kauppinen, M., Savolainen, J., Mannisto, T.: Requirements engineering as a driver for innovations. In: 15th IEEE Int. Req. Eng. Conference, pp. 15–20 (2007)
33. Lyytinen, K., Damsgaard, J.: What's wrong with the diffusion of innovation theory. In: Fourth Working Conf. on Diffusing Software Products and Process Innovations, pp. 173–190. Kluwer, B.V, The Netherlands (2001)
34. Petrenko, A.K., Petrenko, O.L.: Formal methods and innovation economy: Facing new challenges. In: Proc. of the 6th IEEE Int. Conf. on Software Engineering and Formal Methods, SEFM 2008, pp. 367–371. IEEE CS, Washington, DC (2008)
35. Grube, P., Schmid, K.: Selecting creativity techniques for innovative requirements engineering. In: 3rd Int. Workshop on Multimedia and Enjoyable Requirements Engineering, pp. 32–36 (September 2008)
36. Böckle, G.: Innovation management for product line engineering organizations. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 124–134. Springer, Heidelberg (2005)
37. Riepula, M.: Sharing source code with clients: A hybrid business and development model. IEEE Software 28(4), 36–41 (2011)
38. Bayer, J., Melone, N.: A critique of diffusion theory as a managerial framework for understanding adoption of software engineering innovations. Journal of Systems and Software 9(2), 161–166 (1989)
39. Prechelt, L., Oezbek, C.: The search for a research method for studying oss process innovation. Empirical Softw. Engg. 16(4), 514–537 (2011)
40. Copeland, P.: Google's innovation factory: Testing, culture, and infrastructure. In: Third Int. Conf. on Soft. Testing, Verification and Validation, pp. 11–14 (April 2010)
41. Forbath, T., Brooks, P., Dass, A.: Beyond cost reduction: Using collaboration to increase innovation in global software development projects. In: IEEE Int. Conf. on Global Soft. Eng (ICGSE), pp. 205–209 (August 2008)
42. Jansen, S., Brinkkemper, S., Hunink, I., Demir, C.: Pragmatic and opportunistic reuse in innovative start-up companies. IEEE Soft. 25(6), 42–49 (2008)
43. Hanssen, G.K.: Opening up software product line engineering. In: Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering, PLEASE 2010, pp. 1–7. ACM, New York (2010)
44. Yilmaz, L.: An agent simulation study on conflict, community climate and innovation in open source communities. IJOSSP 1(4), 1–25 (2009)

45. Carrero, M.: Innovation for the web 2.0 era. Computer 42(11), 96–98 (2009)
46. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly 28(1), 75–105 (2004)
47. Wieringa, R., Daneva, M., Condori-Fernández, N.: The structure of design theories, and an analysis of their use in software engineering experiments. In: Proc. 5th Int. Symp. on Empirical Software Engineering and Measurement, pp. 295–304. IEEE (2011)
48. Wnuk, K., Pfahl, D., Callele, D., Karlsson, E.A.: How can open source software development help requirements management gain the potential of open innovation: an exploratory study. In: Proc. of the ESEM 2012 Symposium, pp. 271–280. ACM, New York (2012)
49. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Natt och Dag, J.: An industrial survey of requirements interdependencies in software product release planning. In: 5th IEEE Int. Symposium on Req. Eng., Toronto, Canada, pp. 84–93 (2001)
50. Wnuk, K., Callele, D., Regnell, B.: Guiding requirements scoping using roi: Towards agility, openness and waste reduction. In: 18th IEEE International Requirements Engineering Conference, Sydney, Australia, pp. 409–410 (2010)
51. Williams, L., Maximilien, E., Vouk, M.: Test-driven development as a defect-reduction practice. In: 14th Int. Symp. on Soft. Reliability Eng., pp. 34–45 (2003)
52. Regnell, B., Runeson, P.: Combining scenario-based requirements with static verification and dynamic testing. In: 4th Int. Working Conference Requirements Engineering: Foundation for Software Quality, pp. 195–206 (1998)
53. Engström, E., Runeson, P.: A qualitative survey of regression testing practices. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 3–16. Springer, Heidelberg (2010)
54. Engström, E., Runeson, P., Skoglund, M.: A systematic review on regression test selection techniques. Information and Software Technology 52(1), 14–30 (2010)
55. Karlström, D., Runeson, P.: Integrating agile software development into stage-gate managed product development. Emp. Soft. Eng. 11(2), 203–225 (2006)
56. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology 50(9-10), 833–859 (2008)
57. Oručević-Alagić, A., Höst, M.: A case study on the transformation from proprietary to open source software. In: Ågerfalk, P., Boldyreff, C., González-Barahona, J.M., Madey, G.R., Noll, J. (eds.) OSS 2010. IFIP AICT, vol. 319, pp. 367–372. Springer, Heidelberg (2010)
58. Karlström, D., Runeson, P., Nordén, S.: A minimal test practice framework for emerging software organizations. Soft. Testing, Verification and Reliability 15(3), 145–166 (2005)
59. Robinson, B., Francis, P.: Improving industrial adoption of software engineering research: a comparison of open and closed source software. In: Proceedings of the ESEM Conference. ACM, Bolzano (2010)
60. Runeson, P., Höst, M., Rainer, A.W., Regnell, B.: Case Study Research in Software Engineering. Guidelines and Examples. Wiley (2012)
61. Schmidt, S.: Shall we really do it again? the powerful concept of replication is neglected in the social sciences. Review of General Psych. 13(2), 90–100 (2009)
62. Cruzes, D.S., Dybå, T., Runeson, P., Höst, M.: Case studies synthesis: Brief experience and challenges for the future. In: Proceedings of the 2011 ESEM Symposium, Banff, Canada (2011)
63. Sjøberg, D.I.K., Dybå, T., Anda, B., Hannay, J.E.: Building theories in software engineering. In: Guide to Advanced Empirical Soft. Eng. Springer (2008)

Improving Businesses Success by Managing Interactions among Agile Teams in Large Organizations

Antonio Martini, Lars Pareto, and Jan Bosch

Software Engineering Division
Chalmers University of Technology, University of Gothenburg
Göteborg, Sweden
{antonio.martini,pareto,jan.bosch}@chalmers.se

Abstract. To achieve successful business, large software companies employ Agile Software Development to be fast and responsive in addressing customer needs. However, a large number of small, independent and fast teams suffer from excessive inter-team interactions, which may lead to paralysis. In this paper we provide a framework to understand how such interactions affect business goals dependent on speed. We detect factors causing observable interaction effects that generate speed waste. By combining data and literature, we provide recommendations to manage such factors, complementing current Agile practices so that they can be adapted in large software organizations.

Keywords: Agile software development, inter-team interaction, speed, large scale software engineering, software business.

1 Introduction

Large software industries strive to make their development processes fast and more responsive with respect to customer needs, minimizing the time between the identification of a customer need and the delivery of a solution. An open issue is how to scale Agile Software Development (ASD) from successful small software projects [1] to large software companies. One successful approach is to split the products in components and features and to parallelize the development using small, fast teams [3]. However, such approach brings the drawback that a team requires interaction with many other teams [19]. The support for such interactions with a high number of teams or with the surrounding organization in Agile methods is weak and not well explored [13]. Some studies highlighted how interaction issues often cause inefficiencies [2],[6],[24],[22], and hinder the speed benefits gained by the parallelization of the development [24]. Also, delays in interaction due to synchronization may turn fast individual teams into slow and frustrated teams constantly forced to wait for others, hindering the fast release of features [3].

There may be several reasons why the teams lose time to interact and to carry out tasks related to such interaction. This speed waste decreases their *interaction speed* and therefore their overall speed.

The purpose of this study is to identify the drawbacks of ASD employed in large-scale software companies related to interaction speed and their impact on business goals depending on speed.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 60–72, 2013.

The research questions addressed in this paper are the following: in the context of large scale ASD,

RQ1 What is inter-team interaction speed?
RQ2 How does inter-team interaction speed affect software business?
RQ3 What factors influence negatively inter-team interaction speed?
RQ4 How can a practitioner detect and manage such factors to increase inter-team interaction speed?

The paper investigates these questions through a multiple-case case-study with three software companies employing large scale ASD. We conducted exploratory group interviews, followed by qualitative data analysis, and member checking sessions.

Our contributions are:

- We define a notion of *interaction speed* as an externally visible property of *organizational boundaries*.
- We describe the *impact* of interaction speed on *business goals* dependent on *speed*.
- We identify factors associated with ASD that cause effects with negative influence on inter-team interaction speed.
- We provide recommendations based on the interviews and the exiting literature.

The paper is organized as follows. In section 2 we outline existing literature on this subject, section 3 describes our theoretical framework that defines our notion of interaction speed. Section 4 describes our research design, section 5 our results (factors, effects and the mitigation practices). Then we discuss the applicability of our results to reach business goals (Section 6) and the limitations of the study (section 7). The paper ends with our conclusions, in Section 8.

2 Literature Review

When surveying the literature, we have found a constant dilemma between, on the one hand, the need to create fast and independent Agile teams [3] and, on the other hand, the need to increase inter-team communications [12]. We found important to understand what interactions are affecting speed and what the causes are. Researchers in Global Software Development have studied interaction problems with the focus on geographically distributed teams [13,12,18,19]. Recommendations found in [12] (*Optimally Splitting Work across Sites, Increasing Communication, Finding Experts, Awareness*) have shown to be important also in large organizations that are considered co-located. This suggests that in large software organizations, even if co-located, the size of the project creates some of the effects as the geographically distributed teams. Therefore, our research may be of value for GSD and vice-versa. In [7] the authors studies how knowledge management affects the coordination of teams. A critical characteristic of ASD in interactions is the informal communication: it has been considered of value for managing volatile requirements, which makes the development flexible but creates challenges for inter-team communication and coordination [17]. In [18] informal communication is suggested as working well for XP with a strong bridgehead between the teams.

A socio-technical framework for evaluating technical and work dependencies has been studied in [4]. However, such framework heavily relies on artifacts representing the ongoing interactions among the employees, which requires reliable artifacts. Such artifacts are not available and are not representative in an informal ASD environment. Thus we found important to understand the interactions in such an environment.

Most of the abovementioned works study only parts of the problem from several perspectives. In [24] the focus is similar to ours, but the studied impact of Agile practices in communication is not related to speed and business goals. Our framework provides a set of factors-effects affecting speed, and recommendations to handle such factors that we haven't found in literature. The research in social psychology presents an opposite perspective on speed and interactions, in which time boundaries are claimed to influence group performance and interaction process [15].

3 Theoretical Framework

Our theoretical framework is based on an initial a priori framework, which has been continuously updated during the study and the analysis of the data [5].

We define *speed* (borrowing the concept from kinematics) as the amount of the *delivered value* (DV) divided by the *value delivery time* (VDT): the time between the perception of a need and delivery of value by some external party. (See fig. 1). VDT is divisible into the time to identify the party (t_N), time until call for action (t_A), the time to commitment (t_C), and the time to delivery of value by the external party (t_D). We recognize the special case of *end-to-end speed*, where the need is perceived by a customer and the value delivered by a supplier.

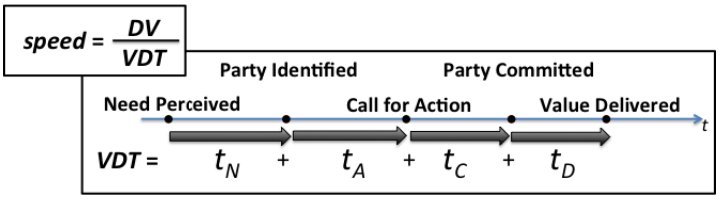

Fig. 1. The definition of speed

A company that seeks to optimize its return of investment of R&D (ROI of R&D) must manage three end-to-end speeds (Fig. 2). The speed with which customer needs lead to new product offers (*1st Deployment speed*), the speed with which new features are replicated in new products (*Replication speed*), and the speed with which change request to an existing product are realized (*Evolution speed*).

End-to-end speeds, in turn, depend on *interaction speed*: how fast teams (or other organizational units), resolve each others' needs. (Fig 3).

Interaction speed relates to both *inter-organizational interaction* (e.g., between teams at the customers and the client side) and *intra organizational interactions* (e.g., between a product management team and a team in a design unit). Each interaction involves several sub-interactions that address sub-needs, and also third party teams.

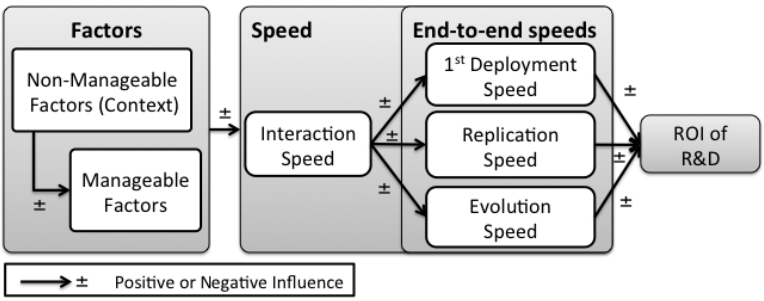

Fig. 2. Three kinds of end-to-end speed and their dependency on interaction speed

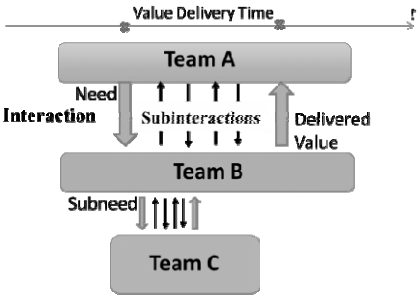

Fig. 3. Interactions, sub-interactions, and interaction speed

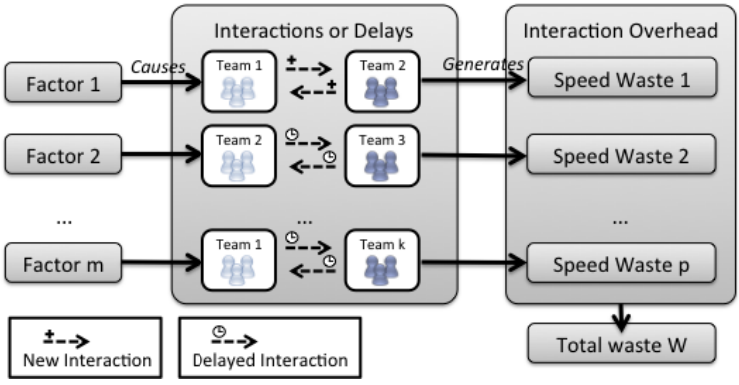

Fig. 4. Factors creating or influencing interactions generating Speed Waste

Interaction speed depends on a number of organizational, architectural, and individual factors that may or may not be managed (Fig. 2). To optimize ROI of R&D we must 1) understand what these factors are, and 2) find strategies to manage them.

In this paper we focus on the impact of interactions on VDT (we assume to have a fixed DV to be delivered, i.e. a set of features). We say that a factor generates *speed waste* if it is either *causing* or *delaying* interaction, which in turn increments VDT decreasing speed (Fig 4). The *total speed waste* (W) is the sum of all such speed wastes.

4 Research Design

We planned a multiple-case case study with engineers and managers in product developing organizations. Our unit of analysis is the *cross functional agile team* and the phenomena of interest the *interaction* speed from the perspective of such teams.

Case Selection: To bring out the complexities of interaction, organizations were to be product-developing companies, with significant maintenance activities, and at least 100 developers. The companies studied were to have several years of experience of ASD. The cases chosen were three large companies with extensive in-house embedded software development. All were situated in the same geographical area (Sweden), but they were active on different international markets. For confidentiality reasons, we will call the companies A, B and C.

Case Description: Company A was a manufacturer of telecommunication systems product lines. The customers receive a platform and pay to unlock new features. The organization was split in cross-functional teams, most of which with feature development roles. Some of the teams had special supporting roles (technology, knowledge, architecture, ect.). Most of the teams used their preferred variant of ASD (often Scrum). Features were developed on top of a reference architecture, and the main process consisted of a pre-study followed by few (ca. 3) sprint iterations before the feature was deployed.

Company B was a manufacturer of utility vehicles; the team developed a communication subsystem for one of their product lines. In this environment, the teams were partially implementing ASD (Scrum). Some competences were separated, e.g. System Engineers sat separately. Special customers requesting special features drove the business, and speed was important for the business goals of this company.

Company C was involved in the automotive industry. Some of the development was done by suppliers, some by in-house teams following Scrum. The surrounding organization was following a stage-gate releasing model. The team we studied developed in-house software, served some projects with different releasing deadlines.

Data collection: data collection was structured in three phases: initial workshops with participants from A, B and C; focus group meetings; validation sessions for reviewing the results.

In the first phase, we conducted semi-structured group interviews with team members (developers and architects), line managers and process specialists. Interviews included participants with mixed roles and revolved around Figures 1-3.

In the second phase we ran 3 focus groups, one for each company. We studied the phenomenon from the team perspective. We included senior developers, team leaders, architects and testers. In this phase we focused on extracting the main factors that were causing or influencing interaction speed. We ran the focus groups separately for each company. We discussed the problem by using models 1-3, then we asked the participants for situations in which the team was suffering from interaction and finally we injected the information from the previous sessions.

In the third phase, after the data analysis, we ran an interview session for each company for validation purposes. Some of the same participants were involved in this process, to adjust researcher's representation of the data. Finally, a short validation workshop with 2 employees from all the companies was conducted.

Data Analysis: After each session we analyzed the recorded interviews to develop models and first results to be discussed in the following sessions. The analysis of the data was carried out between the phase 2 and 3 and also afterwards, to refine the results. We inductively further developed the initial theoretical frameworks and we populated them with factors, effects and improvement practices emerging from the data. We defined each factor, classified it as either *generating* or *influencing* interaction, classified by polarity, and illustrated the importance of managing the factor to increase interaction and end-to-end speed. We have also extracted some suggestions for improvement practices, although such hypothesis need further research.

Synthesis: On the basis of this analysis and suggestions by the informants in interview data, we formulated mitigation strategies for the factors found. Such factors and the mitigation strategies were reported back to informants, and the feedback recorded, analyzed and incorporated in our results.

5 Findings

In the following we show the factors and the effects distilled from the analysis. We have identified 10 *Root Factors,* all manageable. Each factor produces one or more *interaction effects* that are observable in the company. Such effects (and the factors as well) have a negative influence on interaction speed. We have recognized 8 effects:

Table 1. Effects and their explanation

E1. Long waiting time to comm.	A team has to wait before communicating with other ones. This increases t_N, t_A or t_C (Fig. 1).
E2. Long waiting time for value	As the previous one, but the team is waiting for the realization time t_D needed from the other team to deliver the value.
E3. Intense communication	Each instance of inter-team communications requires a long time. Again, this may influence t_N, t_A or t_C.
E4. Corrupted communication	The information received by the team is insufficient to deliver the requested value. This, in turn, may cause intense communication or high interaction frequency (see E5).
E5. High interaction frequency	The number of interactions between two teams is too high (i.e. it clearly hinders the focus on the current development). An instance of this effect occurs when a member in the team is continuously consulted for his or her knowledge by many other teams. This phenomenon has also been called “backpacking” in the interviews.
E6. High task frequency	A single interaction may require many tasks to be carried out in order to deliver the value.
E7. Heavy interaction tasks	The time t_D is long because of the large amount of time required for carrying out the task to deliver the value.
E8. Corrupted value	A (sub-) value has to be delivered to complete the interaction. However, the received value doesn’t satisfy the need that started the interaction.

The *length* and *frequency* mentioned for characterizing the effects are not defined in detail because of the exploratory nature of the study: we have used "long" and "high" to emphasize what seems to be "too much" from the interviews. The same holds for "high frequency". We couldn't establish a specific threshold for the frequency: however, a higher number clearly corresponds to the increment of VDT.

In the following we list 10 Root Factors. For each, we give a definition and we explain what interaction effects they cause in terms of speed. We also explain how they are connected to ASD and what recommendations we suggest, based of the data and the literature.

Table 2. Root Factors

F1. Knowledge unavailability	If a team doesn't have all the knowledge to develop a feature independently, they will try to interact with an expert outside the team, creating interactions. The may have to wait for the expert to be available. The team may alternatively decide to make assumptions on the answers that lead to redo most of the work. The expertise may encompass different kinds of knowledge, such as domain, product architecture and technical knowledge [7]. This factor is connected to ASD and the trend of defining small and self-sufficient teams: the more independent they are, the more isolated, the less effective inter-team communications might be [16]. **Recommendation R1**: make available part-time experts serving different teams and covering critical knowledge (the most requested one). The idea is to decrease the workload in the actual team that is not related to the critical expertise from the expert and make him an inner consultant serving the other teams. Grouping interactions in a defined time-box would avoid high frequency of interactions. This involves a process of identifying the critical knowledge, allocating time to the expert broadcasting the information of such availability to the teams.
F2. Expert's reputation	If an employee has a high reputation of having a specific knowledge, the person will be contacted often. Reputation is not only based on the real knowledge of an employee, but rather on his or her social reputation. ASD principles value social interactions over formal knowledge, amplifying the effects of this factor on interaction speed (as also hypothesized in [12]). Thus, some experts might be more consulted than others because of their social status: this might unbalance the interactions among the teams.
F3. Unclear requirements	The team receives requirement specifications for the features. They may have two interaction problems: the long waiting time before the team is able to receive the specification, or the continuous interaction for clarification of the requirements afterwards. The two problems are connected, according to the interviewees: the time spent on the feature preparation determines the quality of the specification, which influences the elaboration time by the team. **Recommendation R2:** the time spent on creating requirements and architecture artifacts might be decreased in order to start the development as soon as possible: to counter balance this approach, part-time roles of architects and product owners should be established in order to provide constant support to the team during development, avoiding the continuous interaction for clarification.
F4. Unexpected Feature Dependencies	Two features may be designed to interact with each other through APIs or through a component. In some cases, dependencies pop up unexpectedly, e.g. due to indirect (software) interactions or because of socio-techincal reasons (as studied in [4]). The team needs to negotiate APIs or to frequently merge changes on a shared component. The dependencies problem is not covered by any known Agile practice. **Recommendation R3**: In this case, as in other kinds of team (see F3), a brigdehead between the two teams would help coordination. Face-to-face communication is infact beneficial as highlighted in [11]

Table 2. (*Continued.*)

F5. No co-location	Large organizations are forced to spread teams in space. According to our interviews, even the distance of one floor makes them distributed, with consequent delays and lack communication and commitment. **Recommendation R4**: The interviews suggest that the teams that have to interact more intensely should be located closer. It can be considered as another level of co-location with respect to intra-team co-location. Even if something like "inter-team co-location" is not mentioned by ASD per se, it can be considered an extended version of the intra-team co-location suggested by the Agile principles. There are also attempt in literature to mitigate this factor in GSD, e.g. [23],[8]
F6. Lack of common time	Teams may need to synchronize in meetings, which requires common available time. If a team decides to not allocate time for interaction or the allocated time-slots don't match, there is a lack of communication or long waiting times. Causes may be the different locations, different time zones (or with different slots of working hours), calendar interferences or low prioritized interaction. This has also been highlighted in [12]. **Recommendation R5:** some agile practices, such as SCRUM, include support for meetings between SCRUM masters. However, other kinds of programmed available time could be considered, e.g. as mentioned in [14]. **R6:** Also shared calendars would help provide better alignment [12].
F7. Mismatch of team's styles of communication	Different teams may have different "styles" of communication, which may cause delays: e.g. one team mainly uses e-mails and doesn't want to meet in person, whilst the other doesn't reply often to e-mails and is used to communicate through face-to-face meetings. The effect is a lack of communication. Another issue may be the different uses of knowledge containers such as boundary objects (e.g. wikis). The Agile culture of letting teams have their customized processes somehow encourages this mismatch. **Recommendation**: inter-team interfaces between team that need interactions should be improved, for example, with bridgeheads, employees having a strong influence in more than one team ([18] and **R3**). This could also affect the study and the composition of the teams: each team would require the presence of someone socially connected to another team that requires a lot of interactions. Also **R2** may be applied, if architecture or product management teams are involved. Some other practices can be found in communication literature, e.g. [11], [23].
F8. Slow resource indexing	When a member of a team needs to interact, he or she needs to find the correct person or team to interact with. The time spent on such activity (t_N, Fig. 1) may be long and therefore delaying. The informality suggested in ASD seems to work as an amplifier for this factor. The choice of consulting people over formal documents creates "Backpacking" (see E5).
F9. Low prioritized interaction	Once an interaction is needed, the involved parts (single employees or whole teams) have to prioritize the interaction as an on-going task. If the interaction is considered as "low priority", the team will delay tasks and communication, hindering the other team(s) involved. **Recommendation R7**: Tools for creating awareness would help in the understanding the overall situation of the involved teams [12]. Again, the presence of people also connected to other teams would enhance commitment (**R3**, **R2**).
F10. Inter-personal conflicts	Two employees in different teams (or even the whole teams) may consider each other "enemies" (for personal or political reasons). Interactions between these employees may be strongly hindered by delays and corrupted information. Again, a work environment strongly built on social interactions may amplify this factor. Some recommendations for this factor have been suggested on different levels in [10]. However, some social aspects in software development and related (agreed) guidelines need further research [10],[21]. The authors in [20] also suggest that these conflicts may be rooted in unclear requirements (F7).

Finally, we can see, in table 3 below, which factor is responsible for which effect.

Table 3. Factors causing observable effects with negative influence on interaction speed

Manageable Factors	Effects							
	E1 *Waiting comm.*	**E2** *Waiting value*	**E3** *Intense comm.*	**E4** *Corrupted comm.*	**E5** *High inter. freq.*	**E6** *High task freq.*	**E7** *Heavy tasks*	**E8** *Corrupted value*
F1 *Knowledge unavailability*	x	x		x	x	x		x
F2 *Expert's reputation*			x		x		x	x
F3 *Unclear requirements*		x	x	x	x			x
F4 *Unex. Feat. Dependencies*			x		x	x	x	
F5 *No co-location*	x	x		x				
F6 *Lack of common time*	x							
F7 *Mismatch of comm. styles*	x		x	x				
F8 *Slow resource indexing*	x			x				
F9 *Low prior. interaction*	x	x		x				x
F10 *Inter-personal conflicts*	x	x		x				x

Cause ➔

6 Discussion

In this section we explain why our results are relevant for the software business, how the factors can be discovered and how recommendations can be applied by practitioners (managers or teams).

As explained in [22] and in Fig. 2, there are three end-to-end speeds influencing Return of Investment: 1st deployment speed, replication speed and evolutions speed. Interaction speed affects all of them, as explained in the following paragraphs.

- *1st Deployment Speed:* when a set of features is released for the first time, the speed is affected by the interaction speed among the teams that have to integrate the features. This kind of speed helps *hitting the market fast* to anticipate the competitors. Fast deployment speed also *shortens the loop in market testing*.
- *Replication Speed:* when a feature is embedded in a previous release, interactions are needed between the team responsible for the new features and the teams that had developed the former ones. Replication increases ROI when the effort made for the 1st deployment speed is spread on the *release of new products and services based on the existing software.*
- *Evolution Speed:* when a feature needs to be changed after its release, such changes will affect other features, requiring interactions again. The speed in *reacting upon a change request* can be critical for gaining the trust of the customers.

Managers want to reach the abovementioned business goals. Delays over the schedules may be due to speed wasted in interactions. Managers may recognize it but they need the team(s) to observe the effects E1-E8. Since each effect is related to one or more root factors (F1-F10), managers can immediately investigate the status of

such factors in the teams to find which one is the cause for the effect. In Table 2, the connections between factors and effects reduce the solution space for the investigating manager, who saves time and resources. In case the factors are recognized, both the team and the manager (depending on the factor) may decide to apply the recommendations (R1-R7). This process is outlined in Fig. 5.

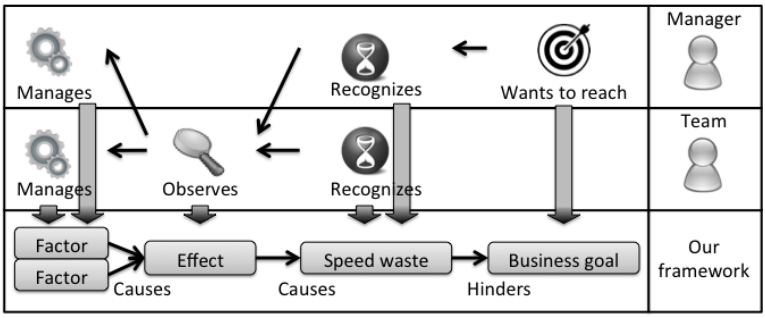

Fig. 5. How the practitioner can use our results

The results reveal that the employment of ASD in large organization brings new root factors hindering inter-team interaction speed and amplifies some of known ones that do not (or limitedly) appear in small projects. Practitioners should be aware of these factors when implementing ASD in large organizations. From the synthesis of the data, the comparison of the factors with Agile principles and solutions from literature, we have summarized a set of recommendations in form of guidelines, which need to be refined by further research, aimed at managing the factors (R1-R7). We have linked the recommendations with the factors (Table 2): this way, whenever an effect is visible, the practitioner could check the corresponding factors and apply the given recommendations (or an adaptation to the company's current situation).

Agile practices (in small contexts) stress the importance of having the customer on site In large companies, the team has to follow requirements defined by a reference architecture and it has to negotiate various kinds of requirements with other teams, i.e. the team has more than one customer: product management, architects and the other teams. Following the "customer on site" principle, all these stakeholders should be available and provide fast feedback to the team. This can be achieved by the creation in the organization of part-time architects, product managers (as in [17]) (R2-R3) and social/formal bridgeheads among interacting teams. Critical expert knowledge should be identified and made available to the teams in form of part-time experts serving multiple teams. As mentioned in [12], a team needs to be aware of the other teams to align with them: shared calendars (R6) and other tools for awareness (R7) would help teams synchronizing. Another solution is mentioned in [14], where "communities of practice" are proposed as programmed events to bring together people from different teams and to spread knowledge (R5). In ASD also the effect of social networks and political dynamics may be amplified, so interacting teams have to be placed close or to mitigate conflicts [10],[13] (R4). Also, the social capital [9] of the organization has to be carefully developed and maintained (for example by defining competence models [25]) in order to increase interaction speed.

7 Threats to Validity and Limitations

In this section we list and explain the limitations for this study. The factors F1-F10 and their recommendations R1-R7 have been analyzed in terms of interaction speed. Other impacts have not been taken in consideration. The information is based on employees' statements and may be biased. The causality of the factors is a hypothesis, and it's not supported by quantitative data yet. The practices are hypotheses synthesized from interview and have been validated in the last session of interviews, but not with precise empirical measurements. One possible threat to validity is the evaluation apprehension: the employees were interviewed usually in groups, which helped balancing statements. To handle the mono-operation bias we collected data from three companies and in some cases from more the one site. As for background influence, we interviewed various roles, from managers to programmers. We limited the threats to conclusion validity (such as influence posed on the subjects) by injecting the preliminary results only after the respondents gave their statements. The threat to external validity (generalizability) has been limited (but not completely solved) by studying three cases with common attributes: size, development domain (i.e. embedded systems) and introducing ASD. We highlighted the differences in the section about their contexts.

8 Conclusions

The effective implementation of ASD in large companies developing embedded software may be the way for the successful achievement of business goals depending on speed. However, Agile teams need to avoid unnecessary interaction and, when unavoidable, to interact efficiently among them and with the rest of the organization. We have described interaction speed (**RQ1**) through the definition of a set of models to frame it with respect to large organizations employing ASD. The reduction of interaction speed negatively influences three business goals: 1st deployment speed, replication speed and evolution speed (**RQ2**). To increase interaction speed and therefore reach such goals, we provided the practitioners with effects (E1-E8, Table 1) observable in the organization, and the factors causing them (F1-F10, Table 2)(**RQ3**). Finally we have proposed a set of recommendations (R1-R7) to manage such factors in order to complement the practices suggested by ASD (**RQ4**).

Future research includes further strategies for managing the factors and a quantitative study of the effects. The long-term objective may include a measurement system for connecting a quantifiable amount of speed waste with the effects and with specific indicators for the factors.

Acknowledgement. This research has been carried out in Software Centre, by Chalmers and University of Gothenburg, and Industry Partners.

References

1. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional (2000)
2. Bosch, J., Bosch-Sijtsema, P.: From integration to composition: On the impact of software product lines, global development and ecosystems. Journal of Systems and Software 83 (2010)
3. Bosch, J., Bosch-Sijtsema, P.M.: Introducing agile customer-centered development in a legacy software product line. Software: Practice and Experience 41 (2011)
4. Cataldo, M., Herbsleb, J.D., Carley, K.M.: Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In: International Symposium on Empirical Software Engineering and Measurement, ESEM 2008. ACM (2008)
5. Dubois, A., Gadde, L.-E.: Systematic combining: an abductive approach to case research. Journal of Business Research 55, 553–560 (2002)
6. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology 50, 833–859 (2008)
7. Espinosa, J.A., Slaughter, S.A., Kraut, R.E., Herbsleb, J.D.: Team knowledge and coordination in geographically distributed software development. Journal of Management Information Systems 24 (2007)
8. Giuffrida, R., Dittrich, Y.: Empirical studies on the use of social software in global software development – A systematic mapping study. Information and Software Technology
9. Greve, A., Benassi, M., Sti, A.D.: Exploring the contributions of human and social capital to productivity. International Review of Sociology 20, 35–58 (2010)
10. Gobeli, D.H., Koenig, H.F., Bechinger, I.: Managing conflict in software development teams: a multilevel analysis. Journal of Product Innovation Management 15, 423–435 (1998)
11. Gotel, O., Kulkarni, V., Say, M., Scharff, C., Sunetnanta, T.: Quality indicators on global software development projects: does "getting to know you" really matter? Journal of Software: Evolution and Process 24, 169–184 (2012)
12. Herbsleb, J.D., Mockus, A.: An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering 29 (2003)
13. Hossain, E., Babar, M.A., Paik, H.: Using Scrum in Global Software Development: A Systematic Literature Review. In: Fourth IEEE International Conference on Global Software Engineering, ICGSE 2009 (2009)
14. Kahkonen, T.: Agile methods for large organizations - building communities of practice. Presented at the Agile Development Conference (2004)
15. Karau, S.J., Kelly, J.R.: The effects of time scarcity and time abundance on group performance quality and interaction process. Journal of Experimental Social Psychology 28 (1992)
16. Karlstrom, D., Runeson, P.: Combining agile methods with stage-gate project management. IEEE Software 22, 43–49 (2005)
17. Korkala, M., Abrahamsson, P.: Communication in Distributed Agile Development: A Case Study. In: 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (2007b)
18. Layman, L., Williams, L., Damian, D., Bures, H.: Essential communication practices for Extreme Programming in a global software development team. Information and Software Technology (2006)

19. Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., Kahkonen, T.: Agile software development in large organizations. Computer 37 (2004)
20. Liu, J.Y.-C., Chen, H.-G., Chen, C.C., Sheu, T.S.: Relationships among interpersonal conflict, requirements uncertainty, and software project performance. International Journal of Project Management 29, 547–556 (2011)
21. Loureiro-Koechlin, C.: A theoretical framework for a structuration model of social issues in software development in information systems. Systems Research and Behavioral Science 25, 99–109 (2008)
22. Martini, A., Pareto, L., Bosch, J.: Enablers and inhibitors for speed with reuse. In: Proceedings of the 16th International Software Product Line Conference - SPLC 2012. ACM (2012)
23. Pawar, K.S., Sharifi, S.: Virtual collocation of design teams: coordinating for speed. International Journal of Agile Management Systems 2, 104–113 (2000)
24. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J.: The impact of agile practices on communication in software development. Empirical Software Engineering 13 (2008)
25. Saldaña-Ramos, J., Sanz-Esteban, A., García, J., Amescua, A.: Skills and abilities for working in a global software development team: A competence model. Journal of Software: Evolution and Process (2013)
26. Strode, D.E., Huff, S.L., Hope, B., Link, S.: Coordination in co-located agile software development projects. Journal of Systems and Software 85, 1222–1238 (2012)

Current Trends in Employee Recruitment Using the Internet

"You need to have a collaborative hiring process"
Steve Jobs

Elfi Furtmueller

Austrian Science Fund, Vienna, Austria
elfi.furtmueller@epiqo.com

Abstract. Reviewing the literature on innovative online services in context with recruiting, seven main services for applicants emerged: general job board services, advanced job search services, social and business network services, mobile services, advanced content and Web 2.0 services, notification services, and lead user services. The identified services from the literature analyses were compared with the state of the art implemented services on 100 international job boards. While English and German-language job boards follow the same trends in terms of services offered to applicants, the former are more innovative and have higher implementation rates of innovative services than the latter. Social and business network integration is increasingly popular but most job boards do not make use of dynamic content for company profiles, real-job-previews or employee testimonials. Notification services, particularly job alerts, have seen successful implementation.

Keywords: Trends, Service Innovation, Employee Recruitment, Applicant, Relationship Management, Internet, Job boards.

1 Introduction

Recruitment via the Internet has become standard [1, 2], and increasing numbers of applicants turn to online services for career advice and information. The focus of online recruitment has shifted from improving "look and feel" and features to attract more applicants during the 2000s towards innovating services [2]. Online career services must motivate the participation of applicants, which requires a high quantity of relevant jobs and integrated search services. Another emerging concern is the mixed success of e-Recruitment implementation [3]. Given these developments, a greater understanding of effective services for users is needed [4, 5]. Services for applicants have been growing due to advancements in technology, yet the literature on innovative means for recruitment is somewhat limited and dispersed among different scholarly communities, with very little attention paid to online job boards. Accordingly, the purpose of this paper is to review and integrate knowledge about service innovations in e-Recruitment.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 73–78, 2013.

The research consists of two parts: a synthesis of the relevant literature and an empirical study. In order to obtain an overview of the service innovation literature in online recruitment, an extensive literature search was conducted in the databases Web of Science and Scopus using the following keywords: "service innovation" OR "product innovation" OR "innovation" AND "recruit*", "online recruit*", "e-recruit*", "web-based recruit*", "internet recruit*, "digital recruit*, "job board" "job portal", "online career services". The resulting selection was evaluated for relevance, and the final sample was read with a focus on synthesizing the existing knowledge [6]. For the empirical part of the study, 100 online job boards were examined with a focus on identifying and categorizing service innovations provided to applicants. These job boards were selected based on their degree of popularity (i.e. Google search rank), comparability (i.e. no specialised or niche job boards) and other websites' recommendations. Fifty percent use English as the primary language and 50 percent use German. The focus of the analyses was on large and leading job boards covering career services in entire markets or countries. Each of the selected job boards was visited online and the services offered to applicants were then identified and content-analyzed. The following section presents the results of the literature and empirical studies.

2 Results

After reviewing and synthesizing the selected literature, seven main services for applicants emerged. These services were then compared with the state of the art implemented services on the 100 job boards surveyed. Four central services are widely implemented across the analyzed job boards: 1) ability to perform an immediate basic job search without registering; 2) ability to create a user account to access more personalized features; 3) ability to upload or create one or more resumes; and 4) the ability to apply directly to a job offer (see Table 1). Overall, 96 % of all studied job boards offer a basic job search without registration, and none require a paid subscription for job seekers.

Table 1. General Job Board Services

Description of service	Total	English	German
Free job search without login	96 %	92 %	100 %
Ability to register a user account	85 %	86 %	84 %
Upload / create a resume	63 %	72 %	54 %
Apply online through job board	82 %	96 %	68 %

While the job search feature is a key service for applicants across all examined job boards [7], advanced search features such as real-time auto-completion of search queries while typing and location-based services are gaining popularity. Using geographic information, job boards can suggest targeted job ads in the region of the job seeker. Considering recent Web 2.0 trends, social and business network services

are increasingly implemented on job boards, particularly the English language ones (see Table 2). The salient features include: 1) the presence of a job board in social and business networks; 2) social bookmarking; 3) signing up to a job board using existing accounts; 4) the availability of a YouTube channel to post video content; and 5) integrated social network features on job boards. Presence in social networks was particularly well implemented, with 70 % of the job boards on Facebook and 65 % on Twitter, ahead of career networks LinkedIn and Xing (46 %). Social bookmarking tools were less common, with the usage of frontrunner Facebook at 46 %. The option to use social network accounts to import personal data (and save time filling in resume data) was only available on a limited number of job boards (16 %). Only 4 % of all examined job boards offer own social networking functionality such as user profile creation or applicant forums. In this context, job boards should implement privacy controls and offer dual profiles (private and professional) with the job seeker being in full control over who gets to see the content [8].

Table 2. Social and Business Network Services

Presence in social networks	Total	English	German
Facebook	70 %	76 %	64 %
Twitter	65 %	72 %	58 %
Google+	39 %	46 %	32 %
LinkedIn / Xing	46 %	50 %	42 %
Social bookmarking			
Facebook	46 %	58 %	34 %
Twitter	39 %	52 %	26 %
Google+	29 %	44 %	14 %
LinkedIn / Xing	28 %	40 %	16 %
Login using a social network account	16 %	24 %	8 %
YouTube	32 %	34 %	30%
Built-in social networking services			
Creation of user profile	4 %	8 %	0 %
Applicant forum	4 %	4 %	4 %
Public / private profile differentiation	2 %	4 %	0 %
Privacy control settings	4 %	8 %	0 %

About one quarter (23 %) of the job boards offer a mobile-optimized version. For mobile app availability, the (current) two main providers of mobile operating systems were compared. A native Apple iOS application was found in Apple's "AppStore" for 24 % of the job boards, while a native application from the "Google Play Store", could only be found for 16 %. Advanced content and Web 2.0 services use video, flash, HTML5 and other dynamic content to present content such as company profiles, job previews, or professional help [9]. Company profiles usually include information about the company's size, location, products and services, hiring needs, and contact information. Space on job boards for company profiles is increasingly purchased to support services marketing and employer branding activities [10, 11].

Three different types of notification services were identified among the job boards: email newsletters, email job alerts and RSS feeds (see Table 3). Job alerts stand out in implementation popularity. Using this service, the job seeker can set up one or multiple search queries and automatically receive the search results at regular intervals via email, removing the need for subsequent manual searches. 82 % of the selected job boards offered email job alerts, with German-language sites leading (86 %) over the English-language job boards (78 %). Only 37 % of all studied job boards offered RSS feeds, less than half of those offering email job alerts. This is possibly due to lower awareness of and comfort with RSS on the part of job seekers.

Table 3. Notification Services

	Total	English	German
Newsletter	27 %	26 %	28 %
Email Job Alerts	82 %	78 %	86 %
RSS feeds	37 %	38 %	36 %

Lead user service innovations can be used to improve the communication between applicants and recruiters, such as direct-messaging services or implementing talent pools [8, 12]. Here, recruiters may deliver targeted information such as newsletters, information to company events, or access to job postings to selected applicants. The restricted access to these company talent pools may make job seekers feel privileged to be included and encouraged to return to the job boards on a regular basis [13]. Only 2 % of the English job boards studied offer direct messaging tools between job seekers and recruiters, and 4 % offer some sort of talent pool. Other innovative lead user services include calculations of job seekers' market value, job and person fit matching, applicant blogs, employer ranking or location-based job visualization services. Employer ranking and employee testimonials are relatively easy to implement. However, in that a large part of their revenue comes from recruiters/employers, job boards may eventually refrain from implementing employer ranking services in order to avoid risks to company image or the posting of overly negative content, which may lose them paying customers. Hence, continuous monitoring of user content would be important to generate service innovations [14].

3 Conclusion

This paper synthesized the literature on online services offered to job seekers. Seven main services were detected and compared with the implemented services on 100 international job boards. While English- and German-language job boards follow the same trends in terms of services offered to applicants, the former have higher implementation rates of innovative services. Services to keep users active on job boards after they have found a new job are virtually absent [15]. When looking at successful career networks such as LinkedIn or Xing, the vast majority of subscribers have an up-to-date profile which implies a continuous use of these networks. This

shows that sustained use of a job board is encouraged by the integration of social networking characteristics. For a job board to be successful in the long term, continued and steady service innovation has to be a central part of its strategy for attracting job seekers [8, 13]. The future is likely to belong to those providers who rethink the elusive nature of loyalty in hyperspace, and who best understand their users' shared social identity. Once they grasp this they should strive to provide semantic technologies that genuinely enhance users' online experiences in terms of social exchange, self-esteem, privacy, sense of control and playfulness [16]. Given the surge in demand and accelerating competitiveness among job boards to attract and maintain applicant profiles, a sharper understanding of the factors leading to applicant commitment to specific online career services is needed.

Acknowledgements. The author gratefully acknowledges support by the Austrian Science Fund (FWF), project number: J3346. Special thanks to www.epiqo.com for the helpful guidance.

References

1. Ployhart, R.E.: Staffing in the 21st century: New challenges and strategic opportunities. Journal of Management 32, 868–897 (2006)
2. Furtmueller, E.: Using technology in global talent recruitment: Why HR/OB scholars need IS knowledge? Dissertation, University of Twente (2012)
3. Bondarouk, T.V., Ruël, H.J.M.: Electronic Human Resource Management: challenges in the digital era. International Journal of Human Resource Management 20(3), 505–514 (2009)
4. Tate, M., Furtmueller, E.: 201 Service Development as Action Design Research: Reporting on a servitized e-recruiting portal. In: Proceedings of SIGSVC Workshop. Sprouts: Working Papers on Information Systems, 12(35) (2012)
5. Furtmueller, E., Wilderom, C., Tate, M.: Managing recruitment and selection in the digital age: e-HRM and resumes. Human Systems Management 30(4), 1–17 (2011)
6. Wolfswinkel, J., Furtmueller, E., Wilderom, C.: Using grounded theory as a method for rigorously reviewing literature. European Journal of Information Systems 22, 45–55 (2013)
7. Cappelli, P.: Making the most of online recruiting. Harvard Business Review 79(3), 139–146 (2001)
8. Furtmueller, E., Wilderom, C., van Dick, R.: Utilizing the Lead User Method for promoting Innovation in e-Recruiting. In: Bondarouk, T.V., Ruël, H.J.M., Oiry, E., Guiderdoni-Jourdan, K. (eds.) Handbook of Research on E-Transformation and Human Resources Management Technologies, pp. 251–273. Information Science Reference, New York (2009)
9. CedarCrestone. HR Systems Survey: HR Technologies, Deployment Approaches, Value, and Metrics. 14th Annual Edition (2011-2012), http://www.cedarcrestone.com
10. Bartram, D.: Internet Recruitment and Selection: Kissing Frogs to find Princes. International Journal of Selection and Assessment 8(4), 261–274 (2000)
11. Koong, K.S., Liu, L.C., Williams, D.L.: An identification of Internet Job Board attributes. Human Systems Management 21(2), 129–135 (2002)

12. Furtmueller, E., Wilderom, C., Mueller, R.: Online Resumes: Optimizing Design to Service Recruiters. In: ECIS 2010, Proceedings of the European Conference on Information Systems. Paper 53 (2010)
13. Furtmueller, E., Wilderom, C., van Dick, R.: Sustainable e-recruiting portals: How to motivate applicants to stay connected throughout their careers? International Journal of Technology and Human Interaction 6(3), 1–20 (2010)
14. Tate, M., Furtmueller, E., Wilderom, C.: Localizing versus standardizing electronic human resource management: Complexities and tensions between HRM and IT departments. European Journal of International Management (in press)
15. Feldman, D.C., Klaas, B.S.: Internet job hunting: A field study of applicant experiences with online recruiting. Human Resource Management 41(2), 175–192 (2002)
16. Tate, M., Furtmueller, E.: Sustainable business models for services using semantic web components: Insights from the field. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 26–30. Springer, Heidelberg (2013)

Post-deployment Data Collection in Software-Intensive Embedded Products

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. To stay competitive, software development companies need to constantly evolve their software development practices. Companies that succeed in shortening customer feedback loops, minimizing the time between customer proof points and learn from customer usage data will be able to accelerate innovation and improve the accuracy of their development investments. While contemporary research reports on a number of well-established techniques for actively involving customers before and during development, there is less evidence on how to successfully use post-deployment customer data as input to the development process. As a result, companies invest significantly in development efforts without having an accurate way of continuously validating whether the functionality they develop is of direct value to customers once the product is taken into use. In this paper, we explore techniques for involving customers and for collecting customer data in pre-development, during development and in the post-deployment phase of software development. We do so by studying three software development companies involved in large-scale development of embedded software. We present an inventory of the techniques they use for collecting customer feedback and we outline the key opportunities for more effective development and evolution based on post-deployment data collection.

Keywords: Agile development, customer involvement, customer feedback, post-deployment data collection.

1 Introduction

Market uncertainties, competitive pressures, and the constant need for shortened development cycles call for software development practices that are flexible, responsive and adaptive to customers [1, 2]. To respond to this, many software development companies have, since a decade or more, adopted agile practices. In advocating customer involvement and the importance of test-driven development practices [3], agile practices have attracted not only small software development companies, but also companies involved in large-scale development of embedded products.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 79–89, 2013.

However, while many companies have succeeded in applying agile practices and, as a result, leveraged the benefits of customer involvement and continuous validation of functionality before and during development, there are few examples of companies that have succeeded in establishing techniques for continuously collecting customer data and validating software functionality also after commercial deployment of the product. The one exception is the Web 2.0 and the software-as-a-service (SaaS) domain where companies like Microsoft [4], Intuit [5] and others, continuously collect customer and product usage data for continuous improvement of the existing product, and as a basis for new product development. Outside of this domain, there is little evidence that software companies have established techniques for collecting and capitalizing on data that is generated after commercial deployment of the product. As a result, companies invest significantly in development efforts without having an accurate way of continuously validating whether the functionality they develop is of direct value to their customers.

In this paper, we present a multiple case study on three companies developing software-intensive embedded systems. While in different domains, all companies aim at continuous collection and analysis of post-deployment data in order to advance their understanding of their customers and how their products are used. The contribution of the paper is twofold. First, it presents an inventory of techniques used for customer involvement and customer feedback collection before, during and after product development. Second, it presents the key opportunities for more effective product development and evolution by collecting customer data in the post-deployment phase of software development.

2 Background

2.1 Agile Software Development

During the last decade agile development methods have dramatically changed the way software development is performed. Agile methods are characterized by short development cycles, close customer collaboration, rapid feedback loops, and continuous evaluation of functionality through test-driven development practices [3, 6]. In comparison to plan-driven development methods, agile methods operate on the principle of "just enough method" and seek to avoid cumbersome and time-consuming processes that add little value to the customer. Although agile methods differ in details and techniques, overall principles such as 'flexibility', 'working code' and 'customer involvement' lie at the heart of all of them. With practices such as daily stand-up meetings, joint planning sessions and project retrospectives, agile methods offer a range of techniques for facilitating collaboration and customer involvement. Lately, test-driven development has become a core agile practice for facilitating continuous validation of functionality. While the agile principles were initially developed for smaller software development organizations, evidence show that large software-intensive organizations operating in complex global development environments are in the process of deploying agile methods as part of their de-facto approach to software development.

However, while agile practices are conducive to close customer collaboration and continuous validation of functionality in the early phases of development [7, 8], there is less evidence on companies that have succeeded in establishing techniques for continuously collecting customer data and validating software functionality also after commercial deployment of the product.

2.2 Customer Involvement

To involve customers in the development process is not a new phenomenon and it is well elaborated upon in user-centered development approaches such as participatory design [10], cooperative design [11], joint-application design [12] and other similar approaches. In these approaches, customers are actively involved in the initial exploration and problem definition phase, as well as during development to help evaluate proposed solutions and different design alternatives. For pre-development involvement, techniques such as use cases, scenarios, prototyping, stakeholder interviews, joint requirements sessions, joint application design sessions etc. are common. Likewise, techniques such as alpha- and beta testing, observation, expert reviews, prototyping, measuring and different types of use cases and scenarios are efficiently used during development in order to continuously validate that the functionality that is developed is of value to the customers. As can be seen in research on agile methods [3, 13, 14, 15], as well as prominent research on requirements engineering [16, 17], techniques such as user surveys, user scenarios, customer archetypes and similar representations are used to capture generic customer needs for mass-market products [17]. Likewise, large-scale agile development often uses product management as a proxy for communicating customer feedback to the development organization before and during development of the product [19].

With regard to post-deployment techniques for customer involvement, the concept of 'lead users' is often used to reflect close collaboration with innovative customers in order to use their feedback for improvement and innovation strategies [20]. As recognized in this research, close collaboration with pro-active customers can result in new functionality as well as new product ideas. Similarly, the 'software ecosystem' approach is referred to as a way for companies to build a community of developers and involve customers to contribute to the improvement activities that take place after product deployment [21]. More recently, the concept of 'Innovation Experiment Systems (IES), and technologies such as Web 2.0, social network systems and Software-as-a-Service (SaaS), or 'on-demand' software, have provided companies with new opportunities to observe and measure system use [4, 5], as well as to run frequent experiments with customers to identify what functionality they value. As reported in relation to these technologies, techniques such as A/B testing (or 'split testing'), customer surveys run by using software tools, analysis of bug reports and product performance data are the most common techniques for continuous collection of post-deployment customer and product data.

In Table 1, we summarize different techniques for involving customers and collecting customer feedback as reported in previous research. As can be seen in the table, there are primarily techniques for this in the early phases of development.

However, techniques for collecting customer feedback after commercial deployment are scarce. While illustrative examples can be found in the Web 2.0 and Software-as-a-Service (SaaS) domain, where sophisticated mechanisms for post-deployment data collection exist, these are not easily applicable for companies involved in large-scale, often embedded, software development.

Table 1. Techniques for customer involvement and customer feedback collection as reported in previous research

Development phase:	**Development activity:**	**Customer involvement /feedback collection technique(s):**
Pre-development	Exploration and problem definition Requirements engineering	Use cases Use scenarios Prototyping Stakeholder interviews Joint requirements development sessions Joint application design sessions Customer representatives/archetypes
During development	Evaluation and validation	Alpha and beta testing Use cases Use scenarios Measuring Prototyping Expert reviews Stand-up meetings
Post-deployment	Evolution and maintenance Improvement and innovation	Lead users A/B testing (split testing) Customer surveys Bug report analysis Performance data analysis

3 Research Site and Method

3.1 Research Site

This paper presents on-going research based on a multiple case study conducted at three software development companies. While the companies differ in domain and size, they are all in the process of establishing mechanisms for collecting customer usage data in the post-deployment phase.

Company A is a provider of telecommunication systems and equipment, communications networks and multimedia solutions for mobile and fixed network

operators. They offer end-to-end solutions for mobile communication and they develop telecommunication infrastructure components for a global market. The company is currently shifting their development practices towards agile development and the notion of cross-functional teams, Scrum practices and continuous testing is well established. For the purpose of this study, we met with key stakeholders at two different company sites:

- *Site 1:* The first site is involved in the development and maintenance of nodes within the 3G networks. At this site we met with a group of four people involving the head of system and architecture, two system managers and a deputy manager.
- *Site 2:* The second site is involved in the development, supply and support of media gateways for mobile networks. At this site, we met with a group of six people involving two department managers, a support manager, a senior specialist, a product manager and an integration leader.

Company B is a manufacturer and supplier of transport solutions for commercial use. The development organization involves coordination of a large number of teams and is largely dependent on supplier organizations. While the majority of the organization is plan-driven in character, there are parts in which agile practices are well established and in which Scrum and XP practices are common practice. For the purpose of this study we met with two attribute leaders, two developers, and one software expert focusing on change management and process improvement.

Company C 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 transforming their development practices to reduce lead-time, shorten feedback cycles and increase customer feedback in the development process. For the purpose of this study we met with a group of seven people involving the company CTO, two team leaders, a test manager and two software architects.

3.2 Research Method

Our paper reports on a multiple case study [22] involving three companies involved in large-scale development of embedded software products. The main data collection method used was semi-structured group interviews with open-ended questions [23], with groups of four to seven people. In total, four group interviews were conducted with key people from the different companies. All group interviews were conducted in English and lasted for about two hours. During the interviews, we were two researchers sharing the responsibility of asking questions and facilitating the group discussion to make sure that everyone got a chance to give his/her opinion on the topic. Notes were taken during all interviews and after each interview these notes were shared among the researchers to allow for a discussion regarding the interview session, the answers that had been given and the overall impression of the discussion. As a complement, e-mail correspondence with company representatives was used to clarify any misunderstandings in the transcription of the data.

In terms of data analysis, a qualitative grounded theory approach was adopted [24]. A problem that has been identified in relation to qualitative research is that different individuals may interpret the same data in different ways [25]. This problem was 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 used a 'venting' method, i.e. a process whereby interpretations are discussed with professional colleagues [26]. By sharing notes, and by discussing the results of each group interview, we could develop an accurate understanding of the different contexts.

4 Findings

In this section, we present our interview findings. These are summarized in Table 2 in which we present an inventory of techniques used for customer involvement and collection of customer feedback before, during and after product development. Also, we present the key opportunities for more effective product development and evolution by collecting customer data in the post-deployment phase of software development. The key opportunities were identified during our study and expressed as important by our interviewees when reflecting on ways in which post-deployment data collection can help improve their development practices.

4.1 Pre-development Techniques

Our interviews reveal a range of techniques that are used for involving customers in pre-development activities. In company A, customer contact is channeled through customer units that facilitate communication and coordination between development sites and customers. Twice a year, customer unit workshops are arranged in order for developers and customers to meet. At these workshops a number of different activities are arranged with, e.g., product seminars are held, user groups and test groups meet and lead customers host seminars where specific technology and functionality is discussed. The purpose of these workshops is knowledge sharing and our interviewees report on them as very useful. In addition, customer surveys are used on a regular basis to capture customer experience and satisfaction. At both sites, agile methods are used as the de facto approach to software development and during recent years the product owner role has become the norm for representing the customer in the early phases of development. Together with product management, the product owner works as a 'customer proxy' making sure that customer feedback is reflected in the prioritization of features.

In company B, customer surveys and questionnaires are the most common techniques for involving customers before development starts. Also, the opportunity to meet with customers face-to face is used when possible. In company C, development is based on traditional requirements engineering techniques influenced by customer surveys, product seminars etc. The products are sold to a mass-market and the functionality is prioritized based on generic customer needs.

4.2 During Development Techniques

In all companies, the development phase is characterized by regular customer involvement. In company A (site 1), the development organization is organized so that a number of dedicated teams can work closely with selected customers in order to meet their specific needs. In this way, fast response and customized solutions can be achieved although the customer base is large and dispersed. These customer-specific teams started out as an experiment but have quickly become part of the regular development organization representing an opportunity to increase responsiveness to individual customer needs. In all companies, phone- and videoconferences are common techniques to interact with customers during development. Also, all companies report on site visits as important for increasing the understanding of the customer during development. In company B, the development phase is characterized by constant prototyping, proof of concept and test lab activities. The opportunity to let customers try the vehicles, i.e. test drive, is used when possible. During such test drives important data can be collected and there is the opportunity for customers to communicate directly with the product developers. Safety critical systems as well as functionality related to user experience are constantly tested with customers to validate what functionality they need. In company C, customers are involved on a regular basis by using phone and video conferencing techniques. However, in our interviews at company C we learnt that even though customer representatives are common, developers rarely meet or interact with the customers, or end-users, of their products, and the overall impression was that direct customer feedback was often difficult to achieve.

4.3 Post-deployment Techniques

Based on our interview findings, we see that post-deployment customer involvement in terms of collection of customer usage data is scarce. In company A, both sites report on system operation and performance as types of product data being continuously collected. Also, bug report data is collected in order to learn about system behavior and use. Based on this data, statistical analysis and trend analysis is done and there is the opportunity to learn about current system operation and future dimensioning needs. However, while performance data, such as upgrade success and downtime reports, is collected, company A report on difficulties to use the data. As it seems, customer data is used for trouble-shooting and for maintaining the current version of the product, but very seldom for improving functionality or as a base for developing new functionality. Managers at both sites describe a situation in which data is collected but not used, and they find it difficult to analyze the data to, for instance, learn about what features that are used and what features that are “waste”.

In company B, diagnostic data is collected when the vehicle attends service at an authorized garage. Based on this data, data mining techniques are used to learn about product use. However, while this data is useful for the next iteration of development, i.e. for the next version of the product family, it is collected with very long intervals and is not used for improving the current version of the product or as input to existing functionality. Also, to integrate and to visualize the data is regarded difficult.

In company C, there are no established techniques for post-deployment data collection. While large amounts of data are generated in the systems, these are not used to systematically improve current versions of the systems.

In Table 2, we summarize the techniques that are used for involving customers and collect customer feedback in the different phases of development. As can be seen, the companies have a number of techniques for involving customers in the pre-development and development phase. However, and as recognized in previous research, techniques for post-deployment customer data collection are few.

Table 2. Techniques that the case companies use for customer involvement and customer feedback collection before, during and after product development

Company:	**Pre-development:**	**During Development:**	**Post-deployment:**
A (site 1)	Customer unit workshops Product seminars Surveys Customer representatives	Phone conferences Video conferences Customer-specific teams Stand-up meetings Customer site visits	Product data collection, e.g. system operation/performance data Bug report data collection
A (site 2)	Customer unit workshops User groups Test groups Product seminars Lead customer meetings	Phone conferences Video conferences Customer site visits Stand-up meetings	Product data collection, e.g. system operation/performance data Bug report data collection
B	Questionnaires Interviews Face-to-face customer meetings	Proof of concept Prototyping User test labs Test driving	Diagnostic data collection, e.g. trouble codes, failure reports etc.
C	Surveys Product seminars	Phone conferences Video conferences Customer site visits	Product data collection, e.g. frames per second etc.

4.4 Key Opportunities

While still in the process of establishing techniques for post-deployment data collection, all companies view this activity as critical for continuous validation of their development efforts. In our study, we learnt that there are a number of key opportunities associated with post-deployment data collection. The key opportunities were identified by our interviewees and expressed as important when reflecting on ways in which post-deployment data collection can help improve their development practices. As such, these key opportunities represent the main drivers for more

effective product development and evolution by collecting customer data in the post-deployment phase of software development. The key opportunities are:

- To increase accuracy of development efforts by continuous validation of what functionality customers value.
- To improve requirements prioritization based on customer data.
- To design products that allow for post-deployment data analysis.
- To help customers optimize their use of the product.
- To increase the ability to anticipate future customer needs.
- To increase delivery frequency of functionality.

5 Discussion

The notion of customer involvement, and how to efficiently collect customer feedback, has been a topic of intensive research for decades. However, while there is extensive research on customer involvement techniques for pre-development and development activities [18, 27, 27, 29], research on post-deployment data collection is scarce. Recently, companies such as Microsoft [4], Intuit [5] and others, have adopted techniques to collect customer data after product deployment for continuous improvement of the existing product, as well as for new, innovative product development. While this is a promising area for research, there are few examples that go beyond the Web 2.0 and SaaS domain.

In our study, we explore three companies in the process of establishing techniques for post-deployment data collection. Already, these companies have techniques for exploration and problem definition activities, for capturing customer requirements, and for evaluating and validating proposed solutions and different design alternatives. For example, customer-specific teams are used to increase responsiveness, customer units are used for facilitating customer-developer interaction, and roles such as the product owner are applied to enhance customer impact on feature prioritization [30]. For post-deployment activities such as evolution and maintenance, the companies report on different types of operational data that is being collected. However, even though the companies have data collection mechanisms in place, they find it difficult to integrate, communicate and visualize the data so that it becomes accessible for people in their organization. As a result, post-deployment data is only partially used and the companies agree that there is an untapped potential in customer and product data that is collected after product deployment.

While the companies involved in our study show on post-deployment collection of data, none of them use this data as the basis for improvement of the current product or for innovation of new functionality. This shortcoming is recognized in previous research in which concepts such as ‘online experiments’ [4], ‘test-and-learn’ mind-set [9], and development as ‘innovation experiment systems’ [5] is used to denote techniques that use post-deployment customer data to increase the effectiveness of product development efforts. Inspired by these concepts, our interviewees see a number of opportunities associated with post-deployment data collection. These are related to the ability to increase the accuracy of development efforts, to optimize use of the product, and to increase frequent delivery of customer value.

6 Conclusions

In this paper, we highlight limitations in existent research in terms of post-deployment data collection, and the untapped resource that post-deployment customer and product data remains. Based on a multiple case study at three software development companies, we present an inventory of customer involvement and feedback techniques for pre-development, development and post-deployment activities. Our case study findings confirm previous research in that existing examples of post-deployment data collection techniques are few, and that the data that is collected is mainly used for troubleshooting activities but very seldom for improvement and innovation of products. Furthermore, we identify key opportunities for more effective development and evolution by collecting post-deployment customer data.

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 companies involved in the study and the time and engagement allocated by all interviewees.

References

1. Desouza, K., Awazu, Y., Jha, S., Dombrowski, C., Papagari, S., Baloh, P., Kim, J.Y.: Customer-Driven Innovation, Research Technology Management, pp. 35–44 (May-June 2008)
2. Fogelström, N.D., Gorschek, T., Svahnberg, M., Olsson, P.: The Impact of Agile Principles on Market-Driven Software Product Development. Journal of Software Maintenance and Evolution: Research and Practice 22, 53–80 (2010)
3. Highsmith, J., Cockburn, A.: Agile Software Development: The business of innovation, Software Management, pp. 120–122 (September 2001)
4. 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)
5. Bosch, J.: Building Products as Innovations Experiment Systems. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 27–39. Springer, Heidelberg (2012)
6. Hansen, S., Berente, N., Lyytinen, K.: Emerging principles for requirements processes in organizational contexts. Networking and Information Systems 13, 9–35 (2008)
7. Mishra, D., Mishra, A.: Complex software project development: Agile methods adoption. Journal of Software Maintenance and Evolution: Research and Practice 23, 549–564 (2011)
8. 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: Proceedings of the 38th Euromicro Conference on Software Engineering and Advanced Applications, Cesme, Izmir, Turkey, September 5-7 (2012)
9. Davenport, T.H.: How to design smart business experiments. Harvard Business Review (February 2009)

10. Schuler, D., Namioka, A.: Participatory design: Principles and practices. Erlbaum, Hillsdale (1993)
11. Grønbæk, K., Kyng, M., Mogensen, P.: CSCW challenges: cooperative design in engineering projects. Communications of the ACM 36(6), 67–77 (1993)
12. Wood, J., Silver, D.: Joint application development. John Wiley & Sons, New York (1995)
13. Abrahamsson, P., Conboy, K., Wang, X.: 'Lots done, more to do': the current state of agile systems development research. European Journal of Information Systems 18(4), 281–284 (2009)
14. Beck, K.: Embracing Change with Extreme Programming. Computer 32(10), 70–77 (1999)
15. Larman, C.: Agile and Iterative Development: A Manager's Guide. Addison-Wesley (2004)
16. Nuseibeh, B., Easterbrook, S.: Requirements engineering: A roadmap. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland, June 4-11 (2000)
17. Bennett, K.H., Rajlish, V.T.: Software maintenance and evolution. In: Proceedings of the 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland, June 4-11 (2000)
18. Sommerville, I.: Software engineering, 9th edn. Addison-Wesley, Boston (2010)
19. Larman, C., Vodde, B.: Scaling lean & agile development: Thinking and organizational tools for large-scale scrum. Addison-Wesley (2008)
20. Von Hippel, E.: Democratizing Innovation: The evolving phenomenon of user innovation. Journal für Betriebswirschaft 55, 63–78 (2005)
21. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Business Review 82, 68–78 (2004)
22. Walsham, G.: Interpretive case studies in IS research: Nature and method. European Journal of Information Systems 4, 74–81 (1995)
23. Runesson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2009)
24. Corbin, J., Strauss, A.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Sage, California (1990)
25. Kaplan, B., Duchon, D.: Combining qualitative and quantitative methods in IS research: A case study. MIS Quarterly 12(4), 571–587 (1988)
26. Goetz, J., LeCompte, D.: Ethnography and Qualitative Design in Educational Research. Academic Press, Orlando (1984)
27. Beyer, H., Holtzblatt, K.: Contextual design: Defining customer-centered systems. Morgan Kaufmann, San Francisco (1998)
28. Preece, J., Rogers, Y., Sharp, H.: Interaction design: Beyond human computer interaction. John Wiley & Sons, New York (2002)
29. Pressman, R.S.: Software engineering: A practitioner's approach. McGraw-Hill, New York (2010)
30. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice-Hall (2002)

A Model of Commercial Open Source Software Product Features

Florian Weikert and Dirk Riehle

Friedrich-Alexander-Universität Erlangen-Nürnberg
florian@weikert.it, dirk@riehle.org

Abstract. Commercial open source software has become an important part of the packaged software product industry. This paper provides a model of individual product features, rather than full-fledged business models, and their perceived value to customers. The model is the result of a three-iteration study, including interview analysis, literature review and the implementation of an empirical survey. Companies can use the feature model to determine their products and business model..

1 Introduction

Open source software (OSS) – software whose source code is publicly available and which allows modification and redistribution at no costs – represents a new approach in a world where software used to be kept proprietary. Nowadays, OSS is a major player in important areas such as web browsing (Firefox, Chromium), databases (MySQL), operating systems (Linux) and mobile (Android). Especially the success of Android, which runs on 75% of all smartphones shipped in the third quarter of 2012 [1] indicates that the trend of open source has not yet come to an end.

The $1 billion acquisition of open source company MySQL AB by Sun Microsystems (now part of Oracle Corporation) in 2008 is just one example which shows the commercial potential of this trend [2]. Today, major tech companies such as Google[1], Facebook[2] and Apple[3] use and contribute to open source projects, thus further highlighting the economic significance of OSS.

However, there are still companies that follow the traditional closed-source approach by keeping their software proprietary. Their refusal to reveal the source code of their products to the public might be due to the potential risks of "going open source", e.g. the loss of intellectual property or the increased attack surface for possible lawsuits [3]. Additionally, they might not see the commercial potential of OSS which would compensate them for taking these additional risks. These companies are likely to ask themselves: "How can you make money if you give the

[1] http://code.google.com/intl/de/opensource/projects.html
[2] http://developers.facebook.com/opensource/
[3] http://www.opensource.apple.com/

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 90–101, 2013.

software away for free?"[4]. Or, even more worse: "why should a firm further develop a product if competitors can freely appropriate these contributions?" [5].

As outlined in Section 7, prior research addressed this question on the level of business models. However, companies might still be confused which concrete product features can differentiate a free open source product from an commercial offering: "Which features are our customers willing to spend money on?". We address this question in the present paper by generating a model of commercially viable product features of OSS. The model itself is the result of a three-step process including interview analysis, literature review and the implementation of an empirical survey. The contributions of this paper are:

- A hierarchical model of commercially viable product features of OSS products with detailed explanations on how these features can be used and what their characteristics are;
- A survey-based examination of individual product features based on their frequency and perceived importance.

The paper is structured in the following way: Section 2 sets the scope of this paper and provides definitions which are being used throughout the entire paper. Section 3 outlines the research process while Section 4 displays the final model of product features. Section 5 contains findings from the survey concerning the ranking of individual product features. An evaluation of both the model and the quality of the results can be found in Section 6. Related work is reviewed in Section 7. Finally, Section 8 concludes this paper.

2 Scope and Definitions

Open source software is defined as software covered by an *open source license* – a license which is approved by the Open Source Initiative (OSI) and which complies with the "Open Source Definition"[4]. Consequently, source code must be publicly available and both free redistribution and derivative works have to be allowed.

Open source companies are companies generating revenue based on at least one open source software product. Compared to closed-source companies, this puts them in a special situation since their commercial offerings have to compete with an open source product which is available for free. Consequently, these companies have to provide substantial additional value in order to motivate users to pay for their commercial offerings.

This paper considers three different types of open source companies, namely *software producers (vendors)*, *service providers* and *distributors*. Software producers create and sell OSS products, thus capitalizing on their own intellectual property. In contrast, service providers typically offer services such as training or consulting for third party OSS products which are not their intellectual property. Distributors integrate a set of third party OSS products into a configured, ready-to-use product.

[4] `http://opensource.org/osd`

It is important to note that this categorization is based on the dominant aspect of a company's business model. For example, software producers may also offer services for their own products without being considered a service provider. Hybrid business models are beyond the scope of this paper. Additionally, companies offering OSS products in order to sell complementary hardware or to generate revenue through advertisement are also not part of our research.

3 Research Process

The model presented in Section 4 is the result of a three-step iterative process with each iteration resulting in a different stage of the model. While the first two iterations were limited to qualitative research, we employed a quantitative approach in the final iteration in order to evaluate and enhance the model.

3.1 First Iteration: Initial Model

The initial model is based on a set of seven confidential interviews between Dirk Riehle, Anthony I. Wasserman and five employees of open source companies who have been working in the software industry for over 18 years each. They represented a total of three open source companies which have been existing for at least six years at the time of the interview. Furthermore, all three companies are based in the United States and can be categorized as software producers.

We used an open coding approach in order to extract relevant terms from the interview transcriptions. Related codes were grouped together in order to form categories such as *training* or *support*. We used codes within a group in order to find possible dimensions, e.g. *response time* for the category *support* [6].

3.2 Second Iteration: Revised Model

We revised the initial model in three consecutive steps. In each step, we analyzed literature and the product portfolios of open source companies in order to add new features to the model [7]. Literature review also helped us to describe individual features in more detail than we did in the initial model. Additionally, we introduced new categories in order to replace the flat structure with a multi-level hierarchy.

3.3 Third Iteration: Quantitative Evaluation and Enhancements

The last iteration aimed at evaluating and enhancing the revised model. We accomplished this by creating an online survey aimed at employees of open source companies. The questions of the survey can be grouped into four categories:

- First of all, participants were asked to categorize their company based on the definitions from Section 2. This allowed us to analyze answers based on specific business models.

- The second category contained product-feature-matrices where participants had to mark which of the product features was offered in which of their products. Based on these matrices, we were able to deduce how frequently certain features were offered, how features were bundled together and which features were not relevant in practice.
- In the third category we asked participants to rank product features based on their subjective importance. There was one ranking for each feature category such as "support" or "training". We used these rankings in order to compare the subjective importance of each feature with its frequency.
- Finally, we asked for missing features in order to enhance our model.

We used statistical methods to analyze the results of the survey. However, since only 15 valid responses were received, the implications derived from this survey can only be regarded as starting points for future research rather than definitive results.

4 Model of Product Features

The top level of the revised model consists of three major feature categories: legal features, features related to intellectual property and service features. Each of these categories is discussed in detail in this section. Additionally, we will look at every single feature and explain how it is defined, why customers are willing to spend money on it and how its dimensions look like.

4.1 Legal Features

Legal features can be divided into two sub categories: *commercial license* and *permissions*. *Commercial licenses* are important when looking at the so-called "dual licensing" approach. If a company owns the intellectual property rights of a software product, then it can offer its products under multiple licenses. For example, a company can provide its product for free under the GNU GPL license in order to satisfy the condition of being an open source company. Additionally, it can sell the very same product covered by proprietary license, allowing customers to side-step the requirements enforced by an open source license. We identified five major features which might be a reason for such behavior:

Non copyleft Usage Rights. A special type of open source licenses – called reciprocal or copyleft licenses – contains a special requirement: If an OSS product A is covered by a reciprocal license and someone integrates it into his own software B and distributes the resulting product, B has to be licensed under the very same reciprocal license ("viral effect"). Consequently, this implies that B has to be made open source, as well. If someone is not willing to take this step, they might be interested in spending money on a commercial license including non copyleft usage rights.

Warranty. Open source software usually comes without any warranty. Business users, however, might be interested in having a warranty clause in order to mitigate potential

damages. Consequently, warranty can be offered to customers as part of a commercial license. This feature has several dimensions: first of all, it is important to define what exactly is the subject of warranty. Furthermore, warranty period, actions in case of remedy, type of covered damages and limitations such as customer negligence have to be considered as well. These dimensions allow companies to tailor their warranty clause as needed or even to offer several warranty clauses with gradual pricing.

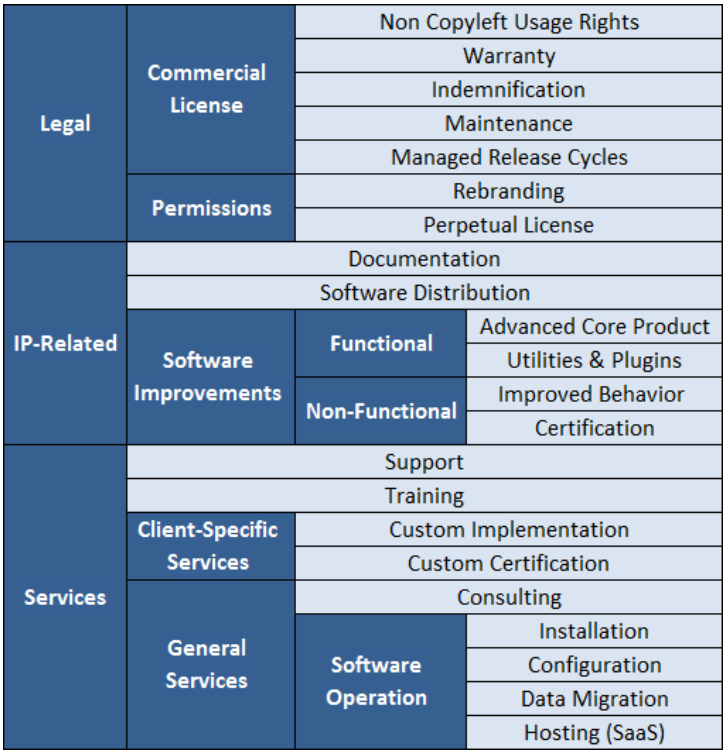

Fig. 1. Hierarchy of product features. From left to right: main categories, sub-categories (both with dark background) and features (light background).

Indemnification. If a customer of an open source company distributes OSS as part of his own products, he can be held responsible for damages caused by the OSS product. An indemnification clause would allow him to transfer this responsibility, thus moving the risk to the original creator of the open source software. Similar to warranty, indemnification means that customers can mitigate potential damages. Consequently, they might be willing to spend money on this feature.

Maintenance. Commercial users typically want access to fast and defined problem solutions for the employed software. Open source users may have to wait for a long time until a bug gets resolved, so commercial users might be willing to pay for a maintenance contract that provides the bug fixes faster and in a way that matches their deployment..

Managed Release Cycles. In contrast to traditional closed-source software, new versions of OSS are released very frequently – sometimes even several new versions per month [3]. While such rapid releases are important in order to ship bug fixes as soon as possible, they also force customers to deal with updates frequently. This is not only time consuming – even more so, it might disrupt customers when significant changes occur often. Especially companies might be interested in receiving fewer, but more stable updates, as one of the interview partners indicated: "Once we're running a production system, you really don't want to have to upgrade and modify it too many times there". Consequently, customers might be charged for such a guarantee.

The second sub category, *permissions*, contains the following two features:

Rebranding. Some software vendors like Openbravo require that their trademarks must not be removed from their open source products [8]. Consequently, even derivative work must display these trademarks. Due to customer perception, however, other companies might like to distribute such derivative work exclusively under their own trademark. This means that they are likely to spend money if they can rebrand the parts of the software belonging to an open source company.

Perpetual License. All interview partners mentioned that their companies use subscription-based payment models instead of charging upfront license fees. As a result, customers can use their OSS products only as long as they pay. However, if they redistribute the OSS product as part of their own product, their customers suddenly depend on the contract between these first-level customers and the open source firm: if those cancelled their subscription, their customers would no longer be allowed to use the derivative software, too. Consequently, they would have to renew the subscription until the last of their customers stops using the derivative software. An open source company can address this problem by selling these customers the permission to grant perpetual licenses to their customers.

It is important to note that several of the above features are completely irrelevant to end-users. In fact, features such as *non copyleft usage rights* and *rebranding* target resellers and OEMs exclusively since they do not provide any value to end-users. Consequently, this distinction has to be considered when defining the legal features of a particular commercial open source offering.

4.2 Features Related to Intellectual Property

This category contains features which are either software or documents to which usage rights are being sold. The top level of this category consists of a sub category and two features.

Documentation. Documents such as reference manuals and user guides are necessary in order to operate and maintain complex software products. Consequently, open source companies can sell (advanced) documentation to the users of their software products.

Software Distribution. This term describes a configured, ready-to-use software product which is the combination of several different OSS products. One prime example is the Linux operating system where several distributions exist, e.g. Ubuntu. It is important to notice that the resulting configuration is the intellectual property of the distributor while the individual components may be owned by third parties [9].

In addition to these two features, the top level also contains the sub category *software improvements*. Typically, open source companies can offer a commercial software product which is based on their OSS product while being somewhat superior. This superiority can be achieved in the following ways:

On the one hand, commercial products can have functional differences by providing functionality which is not implemented in the OSS product. Consequently, the commercial product can carry out additional tasks. Functional differences can be realized in the form of the following two features:

Advanced Core Product. This approach implies that the source code of the open source version is a subset of the commercial version's codebase. Consequently, the functional differences are implemented in the parts of the source code which are kept proprietary.

Utilities and Plugins. Functional differences are realized in the form of proprietary utility applications or plugins which can be used in conjunction with the open source product.

On the other hand, *non-functional differences* are another possibility to differentiate commercial products from their open source counterparts. This sub category contains the following features:

Improved Behavior. This means that the commercial product offers the same set of functions as the open source version does. However, these functions are executed in a superior way. Such qualitative differences can be realized by improving scalability, performance, security, safety, availability, reliability and user experience.

Certification. Commercial versions of an OSS product can be certified for the use with other software or hardware. Furthermore, certification against processes is also possible, i.e. products can be employed in a specific process or their development process meets a certified standard. Customers may be willing to spend money on certification since it guarantees that the product can be operated in the desired way. Moreover, certification may even be a legal requirement in some jurisdictions or where software is employed in highly critical environments.

4.3 Service Features

This category contains the features *support* and *training*, as well as the sub categories *client-specific services* and *general services*:

Support. Similar to maintenance, support is another commercial-only feature ensuring the smooth operation of the software. Although non-paying users can usually receive help through public forums and mailing lists, these options are neither reliable nor do they guarantee a certain response time. Consequently, business users are likely to

spend money on a support contract. Several dimensions of support can be used to design this contract accordingly. First of all, support type and channel have to be defined. One possible option is called "managed support" which implies that customers can interact with employees over phone, email, online chat or official forums. On the contrary, "unguided support" means that customers get access to a set of resources such as knowledge bases and FAQs in order to solve their problems by themselves. Next, customers expect statements regarding the quality of support. Two metrics are of importance: response time and availability. Additionally, open source companies might offer a dedicated support representative to take care of their best customers.

Finally, quantitative characteristics have to be defined. For example, open source companies may limit the number of support incidents if they provide subscription-based support. Additionally, access to the support team can be restricted to a certain number of employees at the customers' companies.

As a result, these different dimensions allow open source companies to create multiple support offerings, each of them addressing the needs of a different customer segment. Consequently, they can sell managed support with 24/7 availability, low response time and unlimited incidents to business users while offering 12/5 support with longer response time and a limited number of incidents to private users.

Training. If the OSS product is sufficiently complex, users might be interested in getting trained on how to use it efficiently. Consequently, open source companies can sell training as a commercial feature. It can be provided online or in real-world class rooms. Online training can either be self-study – by providing documents and online resources - or instructor-led. Additionally, professional certification training can be offered as well. This enables employees to prove that they gained specific knowledge.

The sub category *client-specific services* contains service features which can be requested "on-demand" and whose execution details are particularly tailored to the specific needs of an individual customer.

Custom Implementation. Some customers might have very specific requirements which are not met by the standard software. Consequently, companies can offer to change the implementation of the software or to write additional software components upon the client's request. Since the result of such bespoke services will be superior compared to a general purpose software, customers might be willing to pay for it.

Custom Certification. Due to the great number of available software and hardware, companies are likely to certify their software only against a limited selection of products. If customers want the OSS product to be certified against a very specific product, the open source company might create an slightly changed product which has the requested certification.

The last sub category, *general services*, can be divided into two parts:

Consulting. Similar to traditional closed-source software, open source companies can offer consulting on the use of their products. Additionally, consulting on the specific risks and possibilities of open source software can be provided as well [3].

Software Operation. This term combines services whose sole purpose is to enable the operation of software. For example, companies can perform the installation and

configuration process of their software at the client's office. Furthermore, they can migrate data from legacy systems of the customer to the new software. If an open source company follows a software-as-a-service approach, it can also offer hosting of the software in their own data centers. Consequently, hosting includes installation and migration.

5 Ranking of Features

One of the major goals of the survey was to rank the product features based on their frequency and their importance. Frequency was measured by looking at the number of products containing an individual feature. Additionally, we asked the participants to explicitly rank the features based on their subjective perception of importance. This enabled us to compare both frequency and importance for each feature category separately. As already mentioned in Section 3, the following findings are not representative due to the limited number of only 15 valid answers.

When looking at the category of legal features, we can see that *maintenance* is both the most frequent and the most important feature, followed by *updates*. *Warranty* is the third most frequent feature, although its importance is rather low (sixth place out of eight features). *Indemnification* and *non-copyleft usage rights* can be found in the lower third of both rankings.

Digital documentation and *additional functionality* are the most frequent features related to intellectual property. Furthermore, both features are also the most important ones. The last two spots in both rankings are occupied by *certification for the use in processes* and *certification of development process*. Both features do not appear in any of the recorded products, thus resulting in a frequency of zero.

Although *unguided support* is offered for every product, it is perceived as being the least important feature – *managed support* wins the ranking for the most important support-related feature. In terms of training, *on-site training* is the most frequent and most important feature.

The two remaining categories – client-specific services and general services – show that their most important features are also the most frequent ones. *Custom implementation* wins in the first category while *installation and configuration* followed by *integration* and *consulting* lead the latter.

6 Discussion and Limitations

Our study initially planned to combine exploratory work with confirmatory work. The survey was supposed to validate the models derived from interviews and literature review. The low response rate of the survey (15 valid responses) does not allow us to claim representativeness of the found model and its validation. Thus, the work presented in this paper provides a model of commercial open source features based on qualitative research only. A validation of its correctness is pending and has been left for future work.

How representative then is the presented work? In a still small segment of the commercial open source firms we chose leading companies as best-suited exemplars of their kind, e.g. SugarCRM or Red Hat. In total, we looked at eight commercial open source firms. As is the nature of theory-generating work, these selected exemplars provided deep insight and allowed us to build the model, but we cannot claim to have achieved representativeness, which also isn't the goal of such work.

One of the surprising results of this work is that commercial OSS offerings aren't that much different from traditional closed-source offerings. Most of what commercial open source firms sell has also been sold by traditional firms – except for non-copyleft usage rights. However, open source companies may have a different perspective on some of these features: "[...] closed-source companies are likely to see warranty as a nuisance since it implies additional expenses. On the contrary, open source companies regard such features as a possible way to generate revenue." [7].

Ultimately, the observation about the similarities between the features sold by open source and closed source firms also suggests that our model was able to capture most, possibly all, of the commercially relevant features of open source software products.

7 Related Work

Several papers have addressed the economic relevance of open source software. However, most of them focus on business models as a whole, therefore mentioning concrete product features only as a sideline. This section provides a short description of these features and compares them to our model..

Van Aardt describes thirteen open source business models [4]. By doing so, he also provides possible product features which can be used to generate revenue. First of all, he explains a feature called "packaging" which is similar to *software distribution* in our model. He also addresses *commercial licenses* by looking at the opportunity of dual licensing. Additionally, he describes a feature called "commercial, proprietary software" which is equivalent to *software improvements*. On the service side, he mentions *support*, *training* and *integration* services. Furthermore, he points out that complementary hardware components can be sold.

Some of these services and deliverables are also discussed by Hecker [10]. Additionally, he also mentions features which can be found in our model, including *printed documentation*, *re-branding*, *custom development* and *consulting*. He also discusses complementary online services, which are beyond the scope of our paper.

By classifying 80 open source companies based on their business model, Daffara outlines several product features which are characteristic for specific business models [11]. For example, he identifies a "twin licensing" business model where companies offer non copyleft usage rights as part of their commercial offering. Companies following a "split OSS/commercial products" approach create additional value by offering "proprietary plugins" as part of their commercial products. He also mentions service providers providing features such as training and consulting. Furthermore, he refers to a special type of open source companies labeled "platform providers". In this

context, the term "platform" describes the integration of different open source products. Consequently, it is equal to the feature of "software distribution" in our model.

Fitzgerald identifies two major open source business strategies named "value-added service-enabling" and "loss-leader/market-creating" [12]. These strategies include possible product features such as services (*support*, *consulting*), intellectual property (*software improvements*, *software distribution*) and legal features (*commercial license* with *indemnification* and *warranty*).

As mentioned in Section 2, this paper focuses on three particular open source business models – software producers, service providers and distributors. This categorization is based on a paper by Krishnamurthy in which he also discusses possible product features [13]. For example, he mentions *software updates*, *software distribution* and services such as *support*, *training* and *consulting*.

Riehle identifies four major revenue sources for open source companies and provides an overview of their particular product features [14] [15]. The so-called "core product" refers to a dual-licensing approach, i.e. the open source product is sold with a *commercial license*. On the contrary, companies following a "whole product" approach sell an advanced version of their open source product which has *additional functionality*. Furthermore, companies can provide "operational comfort" by charging for supplementary services such as *support*. Finally, companies can sell "consulting services" such as *training* and *documentation*.

Especially dual-licensing and additional *commercial licenses* are a popular object of investigation. Details and legal implications are discussed by many authors, such as Comino and Manenti [16], Holck and Zicari [17], Lerner and Tirole [18] and Välimäki [19] [20].

8 Conclusions

This paper presents a model of commercially viable products features in open source software, resulting from both qualitative and quantitative research steps. This model provides a hierarchical overview of possible product features and discusses their definition, economic relevance and dimensions in detail. Finally, feedback from an empirical survey is used to enhance the model. Additionally, further details on the relation between individual product features are presented to form the foundation for future work on this topic.

Acknowledgments. We would like to thank the interview partners who kindly shared their knowledge with us, thus enabling us to write this paper. We would also like to acknowledge the contribution of Tony Wasserman, who co-interviewed the commercial interview partners with Dirk Riehle. Finally, we would like to thank the participants of our empirical study for their valuable feedback on the model.

References

[1] IDC, Press Release (November 1, 2012), http://www.idc.com/getdoc.jsp?containerId=prUS23771812#.UOqFc3eSHwx (accessed January 4, 2013)

[2] MySQL AB, Sun Microsystems Announces Completion of MySQL Acquisition (February 26, 2008), http://www.mysql.com/news-and-events/sun/ (accessed January 4, 2013)

[3] Helmreich, M., Riehle, D.: Geschäftsrisiken und Governance von Open-Source in Softwareprodukten. In: Praxis der Wirtschaftsinformatik (HMD 283), vol. 49, pp. 17–25. Jahrgang (February 2012)

[4] van Aardt, A.: Business Models on Open Source Software. In: 19th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2006), Wellington, New Zealand (2006)

[5] Kumar, V., Gordon, B.R., Srinivasan, K.: Competitive Strategy for Open Source Software. Marketing Science, 1066–1078 (November 2011)

[6] Weikert, F.: How To Earn Money With Open Source Software. Friedrich-Alexander University of Erlangen–Nuremberg, Erlangen (2011)

[7] Weikert, F.: Product Features in Commercial Open Source Software. Friedrich-Alexander University of Erlangen-Nuremberg, Erlangen (2011)

[8] Openbravo S.L.U., Trademark Use Guidelines, Openbravo S.L.U. (December 10, 2008), http://www.openbravo.com/legal/trademark-guidelines/(accessed January 4, 2013)

[9] Riehle, D.: Controlling and Steering Open Source Projects, pp. 91–94. IEEE Computer Society (July 2011)

[10] Hecker, F.: Setting Up Shop: The Business of Open-Source Software. IEEE Software, 45–51 (January 1999)

[11] Daffara, C.: Business Models in FLOSS-based Companies. In: s Workshop presentation at the 3rd Conference on Open Source Systems (OSS 2007), Limerick, Ireland (2007)

[12] Fitzgerald, B.: The Transformation of Open Source Software. MIS Quarterly, 587–598 (2006)

[13] Krishnamurthy, S.: An Analysis of Open Source Business Models. Perspectives on Free and Open Source Software, 279–296 (2005)

[14] Riehle, D.: The Commercial Open Source Business Model. In: Nelson, M.L., Shaw, M.J., Strader, T.J. (eds.) AMCIS 2009. LNBIP, vol. 36, pp. 18–30. Springer, Heidelberg (2009)

[15] Riehle, D.: The Single-Vendor Commercial Open Source Business Model. Information Systems and EBusiness Management (2012)

[16] Comino, S., Manenti, F.M.: Dual Licensing in Open Source Software Markets. University of Trient, Trient (2007)

[17] Holck, J., Zicari, R.V.: A Framework Analysis of Business Models for Open Source Software Products with Dual Licensing. In: Copenhagen Business School Department of Informatics, Frederiksberg, Denmark (2007)

[18] Lerner, J., Tirole, J.: The Scope of Open Source Licensing. Journal of Law Economics and Organization, 20–56 (2005)

[19] Välimäki, M.: Dual Licensing in Open Source Software Industry. Systemes d'Information et Management (2003)

[20] Välimäki, M.: The Rise of Open Source Licensing A Challenge to the Use of Intellectual Property in the So ware Industry. Turre Publishing, Helsinki (2005)

A Framework for Strategic Positioning of IT-Products

Wolfram Pietsch

Aachen University of Applied Sciences
Eupener Str. 70, D 52060 Aachen, Germany
pietsch@fh-aachen.de

Abstract. IT Products are viewed and managed differently depending on the perspectives and the stage within the life cycle. A model is presented that integrates different perspectives and stages serving as an aid for the analysis of business models and focused positioning of IT-products. Four generic business models are analysed with regard to the product management function in general and the positioning field for IT-products specifically: off-the-shelf (license), license plus service, project, and system service (incl. cloud computing).

Keywords: Strategic Business Planning, IT Products, Business Models, Product Management.

1 Positioning of IT Products

In order to manage products properly, business issues and technical issues must be dealt with in an integrative manner – product management has been devised as a marketing discipline for this purpose and there is a large body of knowledge (see e.g. [1;2;3]); nevertheless, the major focus is on tangible mass consumer goods. IT-products are not tangible and they consist of different components: besides software, and hardware, also support, training, installation and other services such as consulting and last not least programming. In order manage IT-products properly, the nature of the product must be understood from a business perspective and a technical perspective as well.

Misinterpretations and misunderstandings regarding to IT-products and its management happen quite often in science and practice. For example, an IT Product in may be reduced by some people within an enterprise to the production of a standard DVD for all customers, or it may be viewed by others as a generic set of IT services that must be custom tailored in order to maximise the value for the user. In order to manage IT-products properly, business issues and technical issues must be dealt with simultaneously.

The integration of business and technical issues has been discussed intensively for individual IT Systems culminating in Carr's provocative thesis of the "End of Corporate Computing" [4] which supports the importance of comprehensive IT products and its management. There are several works on software product management from a business view (e.g.[5;6]) and others from a engineering

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 102–116, 2013.

perspective (e.g.[7]). But there is no approach that opens the scope beyond software to IT-product management while integrating business and technical issues. This paper proposes a tool supporting the discourse about the scope of IT products in practice and academia integrating different perspectives such as business and technical. A sound taxonomy is helpful for this discourse, but does not provide decision support. Contingency theory proposes that management approaches may be chosen according to different internal and external factors [6]. Following this approach, the model may lead into a contingency model or IT product management. However, before choosing the 'right' management approach, the situation must be yield a proper segmentation of situations in which IT products must be managed while considering business and technical criteria as well.

The definition of the scope of products with regard to a specific target market is called product positioning within the marketing literature (see e.g. [1;2]) and is crucial for product management in general and for IT products in specific. The IT Product positioning tool described below provides orientation for the management of IT Products, like a compass does for route navigation: it is called the IT product positioning compass.

The Development department of an enterprise may consider any approved development result, which is delivered to a customer, as the IT product. Marketing may differentiate between software licenses, hardware-component, and additional services such as deployment, maintenance and customising. Both perspectives are plausible and legitimate but they should be integrated for effective and efficient processes – to achieve this is the intension of the IT product positioning compass.

The Integration of perspectives is crucial for IT product management. Hence the major critical success factor for successful product management in practice is the persuasiveness of the product manager (see e.g. [2]). Therefore, the IT product positioning compass has been devised to strengthen the consensus dimension of requirements engineering [9]: it should support and improve persuasive skills.

2 'Directions' for the Management of IT-Products

What are the primary "directions" for such an IT-product positioning compass? From a business perspective, the positioning of products is a marketing issue. The four P's of marketing (price, place, graduation, and product) may be interpreted as general 'directions' within marketing worldwide. However, they do not cover IT-related product management aspects and they require some interpretation. The same applies to well-established life cycle models in marketing - they focus on the market perspective solely. The concept of the software life cycle is interpreted differently in the IT industry, nevertheless software life cycle models are well established and may provide 'Directions' for IT Products starting from requirements definition to introduction. Hence, the concept of the life cycle is suitable to determine the primary dimension of the compass.

There is a multitude of different software lifecycle models suitable for different purposes and tailored for different applications. For example, the development of a navigation system must be approached differently than the introduction of a CRM system. There are comprehensive models that may be tailored to different situations, e. g. the German standard 'V-Modell XT' [10]. Such models are very complex and may not be reduced to a single dimension for orientation. Agile models such as DSDM [11] or SCRUM-based approaches [12] are generic and describe an iterative process but not the advancement or operative steps but not general directions. However, software life cycle or agile models are prescriptive and not descriptive: it tells how a software process SHOULD be conducted, but not how it IS conducted typically – leading to the following issues in detail:

1. Life cycle models focus on the development: IT operation and support are important business fields and must be addressed properly.
2. All Stages of the life cycle must be processed step by step: IT Products may span the whole life cycle or only some specific stages.
3. Life cycle models start IT development from scratch: IT-products mostly start with existing soft and hardware components and must be integrated within an existing Infrastructure.
4. The development of different IT-systems may be performed independently: IT Products consist of systems with different life cycles that are heavily intertwined; e.g. the change of an operating system release requires changes in different components and products.
5. Outcomes will be employed in a predefined context: reuse of components is characteristic for IT Products.

Most of the limitations of the life cycle are caused by its procedural interpretation within the systems engineering paradigm. In order to achieve a descriptive model, the stages may not be interpreted as steps but as components, then an IT-Product could cover any stage in an individual sequence: IT Products may cover either development or support, or a sequence of both. This interpretation of the life cycle corresponds to the interpretation of the general product life cycle in marketing.

The stages of a general IT Product life cycle build the primary dimensions of our IT Product positioning compass in order to integrate technical and business perspectives. The stages of the product life cycle are the building blocks for any product. Any IT-Product incorporates development activities within the early stage of the life cycle. In order to be utilised, the IT-Systems must put into operation – two different stages and product types. A Supplier of IT-Products may provide and operate the IT-System or just the software ready for application. Finally there are IT Services and Support that may be part of an IT-Product or may be treated as independent products. The following figure 1 depicts the resulting IT Product positioning compass.

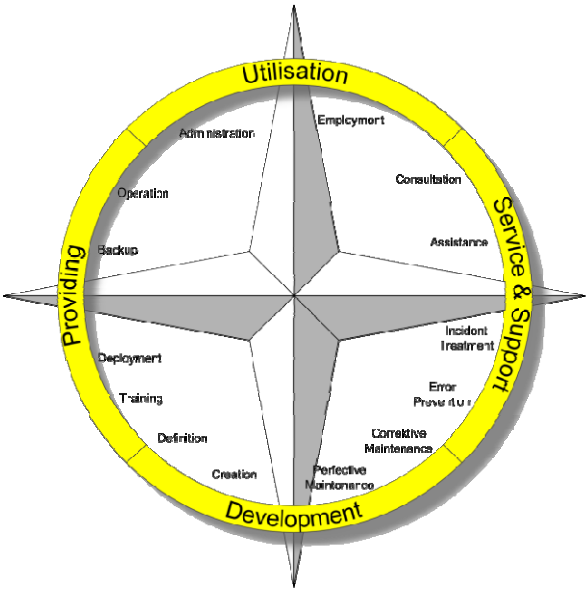

Fig. 1. Directions for the positioning of IT-Products

The four primary directions point out different stages and tasks during the IT Product life cycle that may be incorporated in an IT-Product, such as Programming (Creation), Training, IT-Operation, Consulting, and Maintenance. Supplier of IT-Products may deliver comprehensive IT-products or may specialise in a specific segment. In either case the route must be planned properly: which directions will be approached and in which sequence? These and other tasks must be carried out in any case; hence, there is no specific segment for them in the IT Product positioning compass.

The compass is a positioning tool; it is not geared for the planning of the process, it does not replace the product development plan or process. During the evolution of a product the space of the IT Product positioning compass will be elaborated several times as depicted in figure 2.

Product evolution is never a continuous process; there are several entries, branches and dead ends. The IT Product positioning compass provides a simple framework for analysis and serves as a decision aid: which services are provided currently, which services may be or may not be provided? Does the regular license include perfective maintenance or is an upgrade license required? Decisions about the positioning of products, which are often buried in different project or release plans, are arranged within the IT Product positioning compass into a single picture.

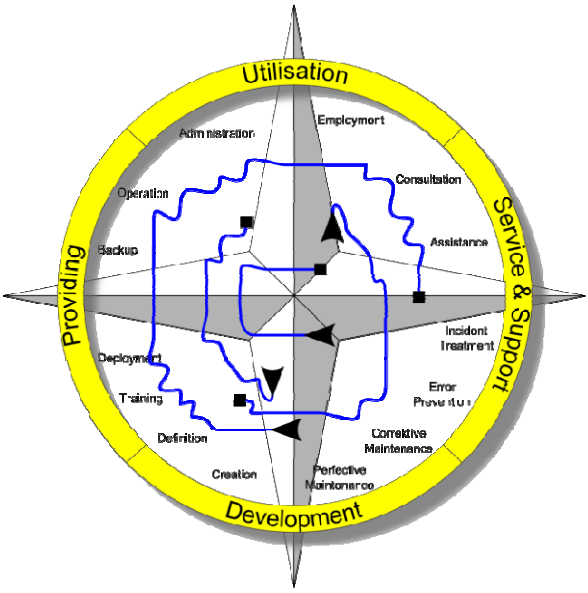

Fig. 2. IT Product Evolution

3 Product Levels

The IT Product positioning compass should improve communication between different roles, i.e. business and technical. One dimension is yet not sufficient for a meaningful positioning. For instance, if database engineers talks with salesmen about maintenance of the product, they may refer to complete different issues. Whereas salesmen are referring to services that are offered to customers, database engineers may address a utility for the reorganisation of data structures. In order to improve product-related communication, it is necessary to clarify the level of the product considering technical and business aspects as well. There are several architectural models that may be employed for this purpose; most of them focus either on the business perspective (e.g. Zachmann's Framework [13]) or on the technical perspective (e.g. the OSI Reference Model [14]). We propose a simple Model that balances business and technology. However, the model may be augmented or replaced by a different model if available and appropriate.

- **Data:** Some IT-products like off-board navigation systems require raw data that is provided by a specialised supplier such as NAVTECH. Hence, there is product level for Data below the technical platform.
- **Technical Platform**: The technical Platform is one level for the management of IT-products. If the product is an embedded system, the IT-Platform is part of the product; for most Software-Products it is not.
- **Elementary Services:** The building blocks of the product that may not be visible to the user are called elementary services. Elementary services are basic functions that may not be useful without other functions, e. g. a backup-function or SMTP.

- **Business Services:** Complex system functions like E-Mail require the combination and calibration of several basic services (SMTP, POP3, E-Mail-Client). Several Business Services may be composed by the same elementary services, e.g. simple and an expert version of the same virus scanner function. On this level, the technical perspective meets the business perspective; business services must provide a benefit that is comprehendible within the application context.
- **Business Processes:** An IT system consists of several Business Services that support or drive specific business processes. At this level, the overall benefit for the specific process(es), task, roles and deliverables is concerned.
- **Value Creation:** Due to limited resources and attention, not every IT System that may provide benefits, will be developed, provided, utilised and supported. The rationale for decisions about IT-systems is addressed at this level ranging from objective criteria such as cost to subjective influences such as personal relationships.

The IT Product positioning compass combines the directions derived from the product life cycle with the levels pointed out above into a polar coordinate system as depicted in figure 3. The resulting framework structures the area of action for the management of IT Products: generation of data, setup of the technical platform, functions, utilisation of data … These tasks require a technical and organizational infrastructure. This infrastructure is a medium but not the target of the IT Product positioning compass. Therefore, is depicted as a secondary dimension. Since the infastructure integrates the IT Product at different levels, it may be located at the edges' of the concentric circles within the IT Product positioning compass.

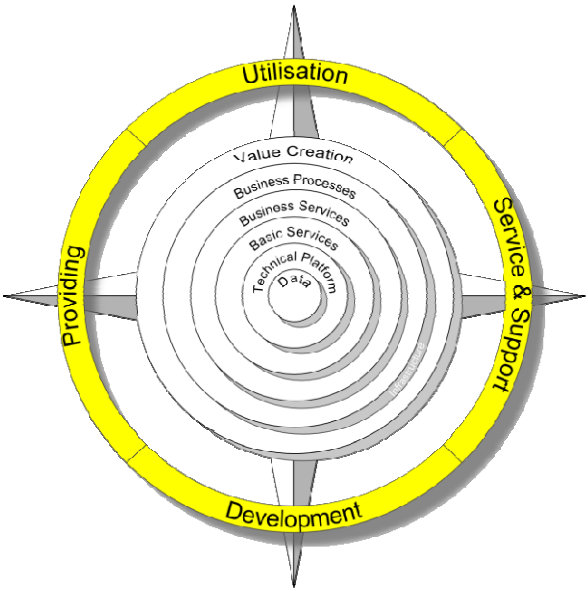

Fig. 3. IT Product positioning compass

4 Positioning of IT Products

4.1 Scope and Objective

In order to employ the IT Product positioning compass for the analysis of product strategies and purposeful positioning of IT-products, the areas of actions must be identified and arranged within the compass space. The following diagram gives a fictitious example of a medium-size software house with a comprehensive portfolio ranging from Consultation, Programming, Introduction, Operations, Help Desk up to Maintenance.

The depicted fictitious software house is offering almost the complete product spectrum. There are unclear boundaries between product areas and it is questionable, whether all areas will be at the same professional level. This product portfolio looks rather like a fuzzy set of services; product positioning is at most unclear. An unclear positioning of the IT-products makes external and internal Communication difficult. Furthermore, it manifests strategic deficits. In practice such deficits often coincides with intensified operative management and very limited product management.

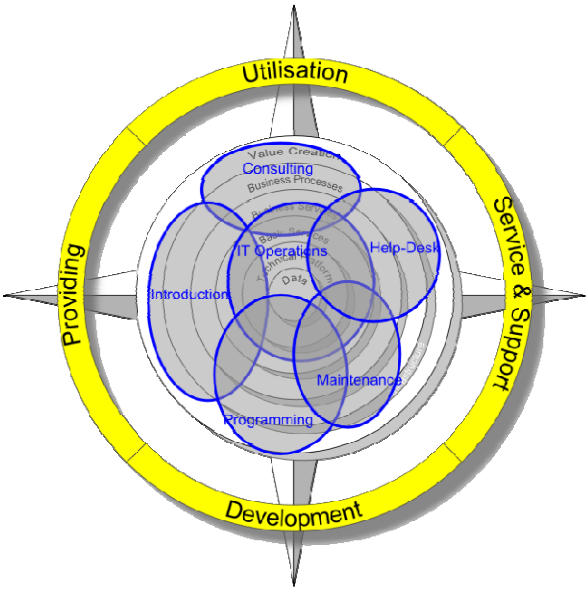

Fig. 4. Fuzzy IT Product Positioning

Product management coordinates and integrates different product-related perspectives, tasks and hierarchical levels in an enterprise [2]; its implementation depends on the specific context. There are several works on specific business models for the software business ranging from an investigation of business culture and critical success factors [15] to an analysis of value chains within the segment of ERP [16]. However, there are no empirical findings available yet on IT Products besides case

studies regarding the context of IT Product management. There several conceptual works ranging from a task analysis for software product management [17] to the definition of generic business models for IT-products [18]. The latter appears to be most suitable for structuring the context of IT-Product Management within the two dimensions discussed above. Four generic business models have been identified for IT-Products: I. Off-the-shelf (License), II. License plus service, III. Project and IV. System services. The IT Product positioning compass will be employed to these four models in order to demonstrate its application for the positioning of IT Products and to enquire into the situative differences for IT Product management.

4.2 Generic Business Models

Type I: 'Off-the-Shelf' (License)

The business model for off-the-shelf products targets at high-volume business. Only specific IT-Products are suitable for this business model, e. g. consumer products like entertainment devices, computer games, or specific utility programs, which spread through the web. Some products are bound to a specific hardware but the software is determining its uniqueness. The buyer acquires the right, to use the system on one ore more devices – the 'License'.

High-volume IT-products require a critical mass of customer as starting point for the marketing. In order to sustain in different market fields, the IT-product must have a certain level of technical maturity and it must not require extensive instructions for usage – it should be marketed 'Off-the-shelf'. In order to achieve high market volume for Off-the-shelf products, the common requirements of the target market are focused, not the specific requirements of single users. Off-the-shelf products are defined by its functions independent from the specific application context. Nevertheless, the potential of the functions to address individual requirements is motivated in terms of general benefits like in the following case: The provision of a dedicated technical interface to PDA' s may enable time-independent work, which is important for field services among others.

The better the functions of the Off-the-shelf product do match the needs and expectations of the target market, the better an IT-Product is positioned. For a small market positioning appears to be easier, but the volume may not be sufficient. The ideal situation is a large homogeneous market and it is one major goal of communication to prepare the market in such a way. For this purpose and many other reasons, Off-the-shelf IT-products should be clearly market-focused and delimited. In general, the transition from development to marketing is critical for product success. Detailed system specifications must be condensed to product definitions that are easy to comprehend, communicable and persistent. There are two specific core competencies: 'Off-the-Shelf'-marketing and 'Off-the-Shelf''-development. It seems to be difficult to integrate both competencies within one organisational unit for high-volume IT-Products. In practice, they are often separated as independent units or even between different enterprises. Marketing and Development of Off-the-shelf IT-products are complementary though self-dependent business models. Development

ensures technical maturity and marketing the marketability. Both models are depicted in figure 5 within the IT Product positioning compass.

Off-the-shelf product require a long term release planning, in order to synchronise development and marketing activities and to give the user planning reliability - an important characteristic of the service quality. Furthermore, maintenance is not only a component of the license but an important mean for product vitalisation. Development is responsible for execution, marketing must analyse the long term market potential.

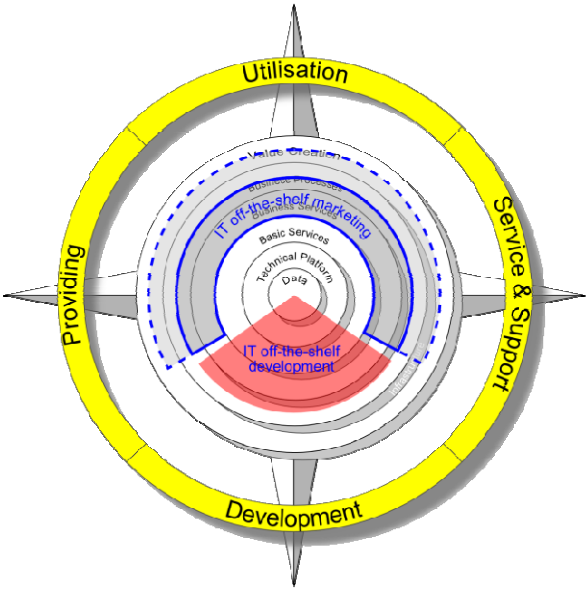

Fig. 5. Positioning 'Off-the-shelf'

Efficiency of institutionalisation and effectiveness of cooperation of marketing, development and eventually product management as an integrative function, determine the success in the off-the-shelf business.

Type II: 'IT License Plus Service'

Only a few IT-product will be marketed just off-the-shelf, there are additional services such as training courses, smaller adjustments etc. If these Services are not mandatory for the employment of the product, they may be considered as an augmentation auf the off-the-shelf business model in order to enrich the product portfolio.

But if an IT-License requires a service in order to be comprehended, deployed or employed, the service is a mandatory part of the IT-Product resulting in a different business model. This applies i.e. for non self-explanatory products, or products that

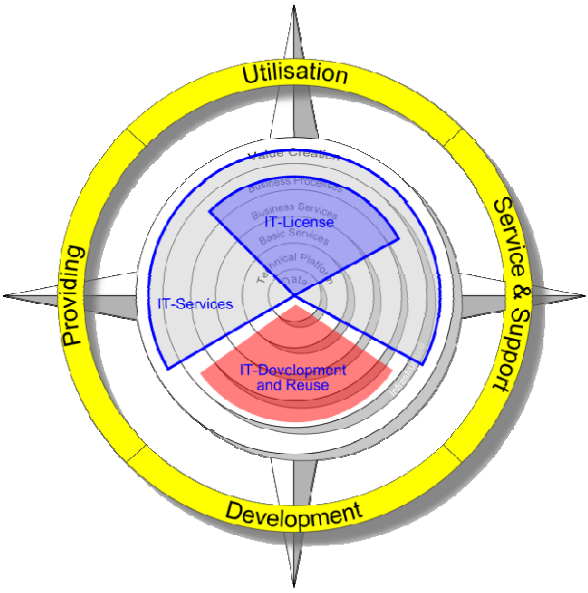

Fig. 6. Positioning for 'License plus Service'

must be adapted to the local infrastructure. The product may not be applied Off-the-shelf, first a feasibility study or proof of concept could be performed, interfaces could be adapted, or there may be intensive training and coaching. The following figure illustrates the positioning range of this business model within the IT Product positioning compass.

The business model 'License plus service' is a combination of a mass product ('Off-the-shelf') and an individual services ('Project'). Whereas Off-the-shelf IT Products may not be modified for an individual customer besides planned parameterisations, individual Modifications may be part of the business model 'IT License plus Service'. There is a standardised core system that may be considered as the 'Off-the-shelf'-part. This core is modified, extended, connected according to individual needs: e.g. regarding the presentation / interaction, reporting or also adaptation specific local interfaces. Typically there are different types of contracts for Licenses and services.

The extent of modifications and the volume of the required services are critical for this business model. If they exceed a certain amount – not more than three person months is a pragmatic rule of thumb – a project organisation is necessary and it is difficult to keep the IT system and services simple and manageable.

License plus service targets at lower volumes than Off-the-shelf; nevertheless, it is based on economies of scales: There are substantial costs to ensure the technical maturity of the core system must. Hence not the single customer is focused, but a larger market segment. This applies to the IT services as well. Services must not be limited a few specialists, but should be organised as standardised processes.

Furthermore, the extent of modifications must be limited in order to reduce complexity. Similar to off-the-shelf products there must be a consolidated product definition, providing the scope of the core system and the limits of the potential functions.

Because of these limitations License plus Service targets primarily at new customers and is very successful in niche markets. Whereas a clear distinction between marketing and development is suitable for the Off-the-shelf-business, both functions must be coupled more closely for License plus Service: The general scope of the core product and individual modifications and must be analysed and managed in an integrative manner by a dedicated product management position.

Type III: 'IT Projects'

The business model License plus Service does not allow substantial modifications or migrations of IT-Systems. Any larger set of individual services must be formalised and executed as an IT-Project. But if there is no IT-system available at the beginning, what is the product? Products require a certain level of maturity and market potential. Hence, the execution of products must meet professional standards and must be tailored to a certain scenario – a specific and proven process is the product.

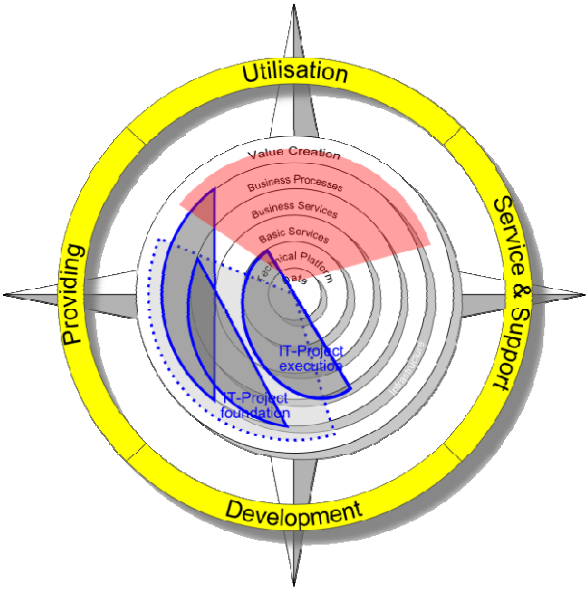

Fig. 7. Positioning of 'IT-Projects'

There is a large body of knowledge regarding the maturity of software development processes [19]. However, product management is interpreted moreover as an administrative function and the marketing function is not addressed at all. In order to establish a viable business model for IT projects, not only the definition and execution of project is crucial, but also its marketing. The maturity and the match of the functions must be ensured for Off-the-shelf products. Likewise, the maturity and the match of the process must be ensured for IT Projects that are marketed as products. The execution of projects is one part of the business model, its foundation the other – both are complementary like those in the Off-the-shelf business model. Figure 7 depicts the positioning of IT-Projects within the IT Product positioning compass.

The scope of IT-Project foundation and execution may vary substantially; indicated by three different segments and one dotted large segment within the figure. IT Project foundation comprises the definition and assessment of processes, the acquisition and qualification of proposals and the administration of resources. Methodically, IT project foundation employs Programme or Portfolio Management techniques and works like a strong IT Project Office: Proposals and projects will be analysed, evaluated, prioritised, initiated, staffed and controlled. Depending upon the target market, personal processes or complex, firmly structured processes will appropriate.

For the marketing of IT-Projects, corporate marketing is highly concerned, since there is no standard system functionality. Firm image, public relation and reference customers are the major levers for project acquisition. Hence Key Account Management and Business Development must be organised properly and coupled closely with, or even performed by project foundation.

Type IV: IT System Services (incl. 'Cloud Computing')

In all business models discussed below, the customer is responsible for the operation of the IT System. Within the business model IT System services operation, administration, comprehensive support and maintenance may be part of the IT Product. In the past, just some niche players have been employing this model successfully. The business model emerged from Application Service Provision (ASP), 'on-demand services' or among others to the general field of Cloud Computing comprised by 'Infrastructure-as-a-Service' (IaS), 'Software-as-a-Service' (SaS) and 'Plattform-as-a-Service' (PaS). Cloud Computing is specific IT Service which employs the Web for the delivery. Since there are other channels for delivery, the more general term IT System Services characterises the business model in general. Figure 8 depicts the positioning range for IT system services. IaS refers to Basic Services. SaaS ans PaS refer to Business Servicess but PaS opens the development part of the circle and has a broader scope.

IT System services may be considered as an extension of the business model Off-the-shelf to the comprehensive IT-System scope of IT-Projects. IT-projects focus on development and introduction, support and maintenance are taken over by the user. Like Off-the-shelf products, IT System services may be utilised by the user immediately and the Development of the IT System must be geared to suit the needs

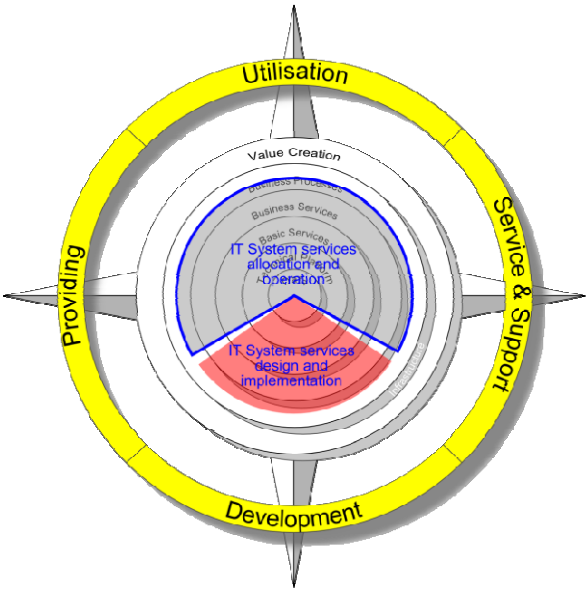

Fig. 8. Positioning 'IT System services'

of many users. Therefore, the business model for IT System services must be split up in two parts analogous to Off-the shelf into design and implementation and allocation and operation (see figure 8). Whereas Off-the-shelf omits the provision of data, technical, platform and basic services (see blank core within figure 5), the technical infrastructure is constitutive for IT System services.

This business model is comprehensive in terms of technology and challenging with respect to marketing: Off-the-shelf products may be comprehended in terms of functionality, Licence plus services are disseminated thru value-added services and Projects thru personal interaction. But the provision of an IT System may not be experienced and/or measured easily. Availability and Performance are common measures for the achieved quality level (→ Service Level Agreements), but they are only indirect measures and do not address the development process and technical and process maturity. Substantial tasks such as data protection, network management are not addressed.

In order to analyse the potential and to position IT System service products, technical criteria and market needs must be balanced within the context of a consistent business strategy [20].

5 Evaluation

Since product management is an integrative task; hence the major critical success factor for successful product management in practice is the persuasiveness of the product manager (see e.g. [2]). The IT product positioning compass has been devised

as an aid for this purpose: it should support and improve persuasive skills. Hence its practical value for IT product managers may indicate validity of the tool if it is analysed properly - this has been the guideline for the invention and design of the IT product positioning compass.

IT product management trainings are carried out by the author and his academic fellows and business partners and associates for an open audience regularly. The authors learned from these training courses that current methods like the product life cycle, portfolio analysis or road mapping are useful for planning purposes, but do not provide orientation with respect to business and technical issues. The authors have been devising an initial version of the compass and it has been employed in a classroom cases, criticised and improved for the next training. The compass has been developed thus in an evolutionary manner until it has reached a state of maturity that has been perceived by the participants as similar to other tools like portfolio analysis. A survey is planned to investigate the dissemination of the in day to day IT product management work.

6 Conclusion

The elaboration of the IT Product positioning compass to four generic business models has been revealing substantial different prerequisites, challenges and pitfalls for the Management of IT-Products. License plus service is a compromise – neither a custom-maid suit nor ready to start. Combinations of the generic business models may be viable, but are not efficient and effective. Different skills, organisation, marketing etc are required for mass products and individual projects. Each business model represents a consistent pattern of strategies and actions. Nevertheless, there could be a coexistence of different models within one enterprise. If Processes and products are clearly separated, the same people could perform task for different product types. However an arbitrary combination may lead to a position ‘stuck in the middle’ in the sense of porter’s generic strategies [21]. Such a position may be accepted as a transition stage or require long-term experience and professionnalism.

The IT Product compass has been devised for IT Product management at first [22]. Nevertheless, is also suitable for the service management design and IT value management. Several cases are in progress and will be published soon.

References

[1] Albers, S., Herrmann, A. (eds.): Handbuch Produktmanagement: Strategieentwicklung - Produktplanung - Organisation – Kontrolle. Wiesbaden, Gabler (2007)
[2] Becker, J.: Marketing-Konzeption. Grundlagen des zielstrategischen und operativen Marketing-Managements. Vahlen, München (2001)
[3] Lehmann, D.R., Winer, R.S.: Product Management. McGraw Hill (2004)
[4] Carr, N.G.: The End of Corporate Computing. Sloan Management Review 46(3), 67–73 (2005)

[5] Conde, D., Condon, D.: Software Product Management: Managing Software Development from Idea to Product to Marketing to Sales. Aspatore (2002)
[6] Dver, A.S.: Software Product Management Essentials. Meghan Kiffer (2003)
[7] Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering. Foundations, Principles, and Techniques. Springer (2005)
[8] Kieser, A., Kubicek, H.: Organisation, 3rd edn., Berlin (1992)
[9] Pohl, K.: The Three Dimensions of Requirements Engineering. In: Rolland, C., Cauvet, C., Bodart, F. (eds.) CAiSE 1993. LNCS, vol. 685, pp. 275–292. Springer, Heidelberg (1993)
[10] Rausch, A., Höhn, R., Höppner, S.: Das V-Modell XT. Springer, Berlin (2005)
[11] Coleman, G., Verbruggen, R.: A quality software process for rapid application development. Software Quality Journal 7, 107–1222 (1998)
[12] Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice Hall (2002)
[13] Zachmann, J.A.: A Framework for Information Systems Architecture. IBM Systems Journal 26(3), 276–292 (1987)
[14] Zimmermann, H.: OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications 28(4), 425–432 (1980)
[15] Cusomano, M.A.: Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets and Manages People. Free Press (1998)
[16] Wolf, C., Geiger, K., Benlian, A., Hess, T., Buxmann, P.: Spezialisierung als Ausprägungsform einer Industrialisierung der Software-Branche – Eine Analyse am Beispiel der ERP-Software von SAP. In: To be published within the Proceedings of the GI-Fachtagung Software-Management 2008, Industrialisierung des Software Managements, Stuttgart, November 12-14 (2008)
[17] van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J.M., Bijlsma, A.: On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. In: Proc. of the International Workshop on Software Product Management, St. Paul / Minnesota, pp. 3–12 (2006)
[18] Pietsch, W.: Geschäftsmodelle als Grundlage für das Software-Produktmanagement. Proc. of the Multikonferenz Wirtschaftsinformatik 2006, Passau, Berlin (2), 211–222 (2006)
[19] Chrissis, B., Konrad, M., Shrum, S.: CMMI. Guidelines for Process Integration and Product Improvement. Addison-Wesley, Boston (2006)
[20] Pietsch, W.: Customer-Oriented Specification and Evaluation of IT Service Level Agreements. In: Richardson, I., Abrahamsson, P., Messnarz, R. (eds.) EuroSPI 2005. LNCS, vol. 3792, pp. 83–94. Springer, Heidelberg (2005)
[21] Porter, M.E.: Competitive Advantage, New York (1985)
[22] Herzwurm, G., Pietsch, W.: Management von Software-Produkten. dpunkt, Wiesbaden (2009)

Cloud Services Pricing Models

Gabriella Laatikainen, Arto Ojala, and Oleksiy Mazhelis

Department of Computer Science and Information Systems
University of Jyväskylä
Jyväskylä, Finland
{gabriella.laatikainen,arto.k.ojala,mazhelis}@jyu.fi

Abstract. A major condition for commercial success is a well-defined pricing strategy, however, cloud service providers face many challenges around pricing. Clearness and transparency in pricing is beneficial for all the actors in the ecosystem, where the currently existing abundance of different pricing models makes decision making difficult for service providers, partners, customers and competitors. In this paper, the SBIFT pricing model is evaluated and updated to cloud context. As a result, a 7-dimensional cloud pricing framework is proposed that helps clarifying the possible pricing models in order to let companies differentiate themselves from competitors by price. The framework can be used also as a tool for price model development and communication about cloud pricing. The taxonomy is based on a broad literature review and empirical research on currently used pricing models of 54 cloud providers.

Keywords: pricing, revenue logic, cloud, SaaS, PaaS, IaaS.

1 Introduction

One of the key conditions for commercial success of cloud services is the clearness and transparency of pricing for both customers and providers [1,2]. Properly applied, a well-defined pricing strategy can change customers' behavior and it can determine the offering's position on the competitive market [3]. 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 [4]. Pricing can also differentiate an offering from the competitors [5,6] and this way increase the company's revenues and position in the market. Therefore pricing is a powerful strategic tool in manager's hands.

However, because of the rapid technology development and increasing competition in the global markets, price modeling for software products became very complex. A number of studies have also suggested that traditional pricing models are not applicable as such for pricing of software products (e.g. [7]) and the way of pricing software products is also changing [8]. Hence, there is a constantly changing labyrinth around software pricing with many different pricing solutions [8]. For this reason, cloud solution providers may face many challenges around pricing [9] and pricing of IT services is often a neglected topic for many IT managers [10].

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 117–129, 2013.

For the above-mentioned reasons, there is a need for a clear and systematic pricing framework, developed especially for cloud industry, that helps decision makers find the proper pricing model and evaluate its alternatives, advantages and disadvantages. Hence, the aim of this study is to examine empirically the applicability of an existing pricing model in the context of cloud solutions and, if needed, propose possible modifications to the model. We seek to contribute to the literature of cloud computing by revealing the most popular pricing models used by 54 cloud solution providers. In addition, we propose a model that managers operating in cloud business can use as a tool to evaluate the proper pricing model for their solutions.

2 Related Work

2.1 The SBIFT Pricing Model

A comprehensive taxonomy of pricing models has been proposed by Iveroth et al. [11], that defines pricing models as systems of price-related characteristics of the agreement between buyer and seller. Price models are described along 5 dimensions, that are listed without priority (see figure 1). According to the authors, price models can be described through the specification of the "positions" on each dimension. The taxonomy is called SBIFT model, that stands for the acronyms of the dimensions.

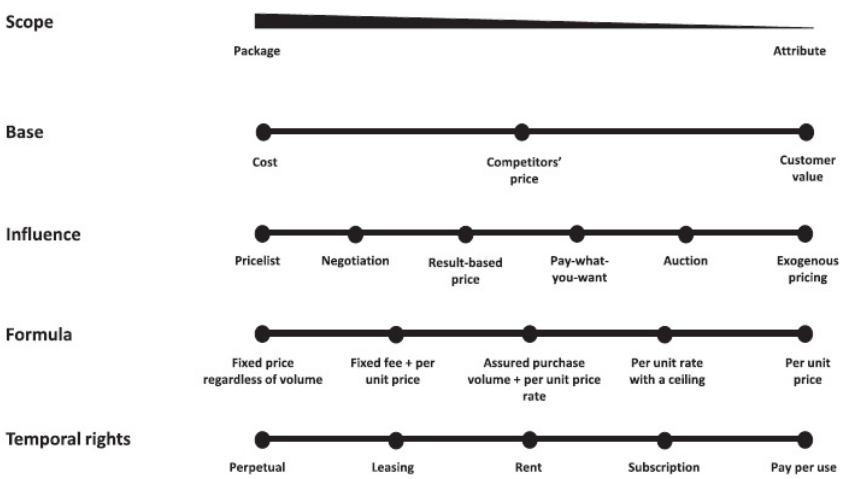

Fig. 1. The SBIFT model [11]

We chose to evaluate this model in cloud context, since it provides the most state-of-the-art and the most integrative work in the current pricing literature. The flexibility of this taxonomy makes it possible to create novel pricing models as a combination of different pricing elements. The model contains pricing elements also from the cloud- and software literature, hence it may be applied to the cloud services easily. The dimensions of the model are presented as follows.

The **Scope** dimension refers to the granularity of the offer. At the left side of the slider, a *Package* of products/services are priced; while the other extreme category is named *Attribute*, referring to the case when each unit of the offer is priced individually and buyers can decide upon buying them or not.

The **Base** dimension refers to the information base that dominates the pricing decisions. *Cost-based pricing* is the most widely used pricing method [12], where the seller determines the price floor based on the cost of developing, producing, distributing and selling the goods. Another pricing formation strategy is setting the price level according to *Competitor's price* of comparable products or services [13]. Using *Value-based* (demand-based) pricing strategies providers define their prices based on the customers' perceived value [10,14,15].

The **Influence** dimension reflects the ability of buyers and sellers to influence the price. If the price is decided by the provider alone, this is usually communicated through a *Pricelist*. If the price is set based on a *Negotiation* between the customer and the provider, then the starting point is also a pricelist but the buyer can influence the final price. The next option is *Result-based pricing*, where the price is determined based on some observable result of the product/service [11]. In an *Auction* the price is set based on the customers' willingness to pay and the sellers' influence on the price is limited. *Exogenous pricing* is used if circumstances beyond the sellers' and buyers' influence determine the price.

The **Formula** dimension refers to the connection between price and volume. With a *Fixed price regardless of volume* (flat-pricing, eat-all-what-you-can), customers pay a fixed price, that is independent from the used volume [16]. The *Fixed fee plus per unit rate* formula has two components: a fixed, predetermined, volume-independent part and a volume-dependent part. In case of *Assured purchase volume plus per unit rate*, a fixed amount of volume is priced with a fix price, and an overage price is charged for the extra consumption with the per unit rate. Using the *Per unit rate with a ceiling* formula, the per unit price has to be paid only until a certain consumption-level, and above that the usage is free of charge [11]. In case of *Per unit price*, units (or units per time) are associated with fixed price values and the customer pays this per unit price regardless of the quality or the economies of scale that the seller might encounter.

The **Temporal rights** dimension refers to the length of the time period when the user can use the offering. In case of *Perpetual* offering, the customer can use and own the goods as long as he wants [17,18,19]. When *Leasing*, customers buy the right to use the service/product for a fixed period and to buy it after the period on a predefined price. Through *Renting* the right is bought to use the product or service for a "rental" period, during which the customer does not get any updates or changes to the original product/service. On the other hand, in

case of *Subscription*, buyers have the right to use the service/product for a period but they also get upgrades, enhancements, new functionalities or new content from the provider during this time. If the buyers pay every time they use the service or product, the seller applies *Pay per use* (pay-as-you-go) mechanism.

2.2 Software Pricing

In software business there are three general revenue models, all including several pricing options. In the first revenue model, software licensing refers to the traditional way to buy the software. In software licensing, customers buy a license that gives right to use the software in a certain amount of computers or processors [17,18]. In many cases, the length or amount of usage is not limited. In the second revenue model, software renting gives right to use the software for a certain time period that is defined in the rent agreement [5]. In the third revenue model, pay-per-use enables software providers to charge customers based on the actual usage of the software [17].

Pricing in these above introduced revenue models may base on different aspects. Lehmann and Buxmann [7] introduced the following pricing parameters:

(i) *Price formation:* The seller determines the price base (cost-based, value-based or competition oriented) and the degree of interaction between the seller and buyer (unilateral or interactive).
(ii) *Structure of payment flow:* Payments may be done as single payments, through recurring payments or through a combination of these.
(iii) *Assessment base:* The number of pricing components, the usage-dependent and usage-independent assessment bases have to be defined.
(iv) *Price discrimination:* Sellers offer the same good to different buyers at different prices. Price discrimination may be first-degree (prices depend on each user's willingness-to-pay), second-degree (customers may choose one of the offered product-price combinations based on required quantity, software version or time), third-degree (market segmentation by the seller based on personal or regional conditions) or multidimensional (combination of these).
(v) *Price bundling:* Several items (services, products, rights, etc.) are bound together into an offering with a predetermined price. The offering may be pure bundling (the products are offered exclusively in a bundle), mixed bundling (goods may be bought as a package or separately), unbundling (products may be bought only separately) or customized bundling (customers choose the content of the bundle). In price bundling, software products, maintenance and support services may be packaged together. The degree of integration of the bundle items can be complementary, independent or they can substitute each other. The price level of the bundle can be additive (the price of the bundle is the sum of the prices of the items), superadditive (the price is greater than the sum of individual prices) or subadditive (lower price than the sum of individual prices).

(vi) *Dynamic pricing strategies:* The seller sets the price dynamically over time. For software products, penetration (setting low prices in the beginning and possibly increasing it later), follow-the-free (the product is free, revenues come from complementary services or extra functionalities) and skimming (high starting prices that may be gradually reduced) pricing strategies are the most important.

Summarizing, the items of SBIFT model [11] and the software pricing parameters [7] overlap each other: some dimensions and parameters refer to the same aspect (Scope-Price bundling, Base-Price determination), some dimensions offer more alternatives than the respective pricing parameter (Influence-Degree of interaction, Formula-Assessment base), one of the dimensions takes a different point-of-view than the respective parameter (Temporal rights-Structure of payment flow) and some parameters are missing from the SBIFT model (Price discrimination, Dynamic pricing strategies).

3 Methodology and Data

In order to evaluate the applicability of the SBIFT model empirically in cloud context and to get an insight into currently used cloud solution pricing models, we studied pricing models of cloud offerings from 54 companies. Our analysis was carried out in September and October 2012 in the following steps: selecting cloud companies for the data sample; search for IaaS-, PaaS- and SaaS-offerings and their pricing information from their webpage; exclusion of those that provide a different type of service or do not provide enough pricing information; evaluation of SBIFT model iteratively. As a result, after searching for pricing data of offerings from more than 160 cloud providers, we could build up 73 pricing models from 54 firms by using the SBIFT model (see Table 1 for more details).

Table 1. Analyzed pricing models

	IaaS	PaaS	SaaS	Total
Number of companies	7	14	33	54
Number of offerings	19	16	33	68
Number of pricing models	20	19	34	73

Data Sample Selection. To ease the search of the cloud offerings, we identified our sample with the help of an internet portal Cloud Computing Showplace[1], that enlists more than 2050 cloud companies. In this online directory, cloud provider companies can register and categorize themselves into IaaS, PaaS and SaaS providers. SaaS providers can also categorize themselves by industry sector and application category.

[1] `http://cloudshowplace.com`

We utilized this portal since it contains the most comprehensive collection of cloud providers compared to other portals (e.g. cloudservicemarket.info or www.saasdir.com) and the number of registered companies are growing continuously, fact that suggests that the directory is an up-to-date, maintained and used portal. To increase the reliability of our sample, we added additional validation steps into the process e.g. by excluding the non-cloud offerings.

We identified our data sample by choosing all registered IaaS and PaaS providers and one SaaS company with relevant pricing data from each industry sector. Since the number of registered SaaS companies is too large and growing constantly, we selected SaaS companies from each industry sector randomly until we had detailed pricing data of at least one SaaS offering from each industry sector in order to increase the industry coverage of the sample data.

Review of the Offerings and Disclosure of Pricing Information. In order to increase the reliability of our data sample method, we reviewed the offerings and excluded the non-IaaS, non-PaaS and non-SaaS services, respectively. Concerning the disclosure of pricing information, our experience is in line with Lehmann et al. [20], who conducted an empirical study on the pricing models of SaaS providers registered on this portal. They found, that especially small and medium size firms provide pricing information on their website. Since not every aspect of the pricing model could be found in most cases, we agreed on excluding data from our sample where the companies did not provide enough information to understand the pricing logic as a whole.

Analysis of the SBIFT Model. During our analysis, we matched each pricing model with a SBIFT pricing model pattern that can be defined as a combination of the positions of the pricing model characteristics along the SBIFT dimensions. While defining the positions, we selected the item that described the pricing characteristic in the most accurate way. The evaluation was done in an iterative process with the following evaluation criteria: (i) Each of the characteristics of the pricing model can be matched to a position of a dimension in the SBIFT model. (ii) One pricing pattern in the SBIFT model describes pricing models, that share the same characteristics. If the evaluation criteria was not met, we modified the SBIFT model to address the problems occurred and started a new iteration until the SBIFT model pattern could be defined for each sample data and the evaluation criteria was met.

4 Research Findings

4.1 SBIFT Model in Cloud Context

Based on our study, we propose some modifications to the SBIFT model that is specific to the cloud services industry (see Figure 2). The framework consists of 7 dimensions depicted in continuous scale, that describe the details of the offering. Next the proposed modifications are described compared to the SBIFT model.

Scope Dimension. Our study revealed that identifying the level of bundling in the Scope dimension is challenging without some kind of categorization between

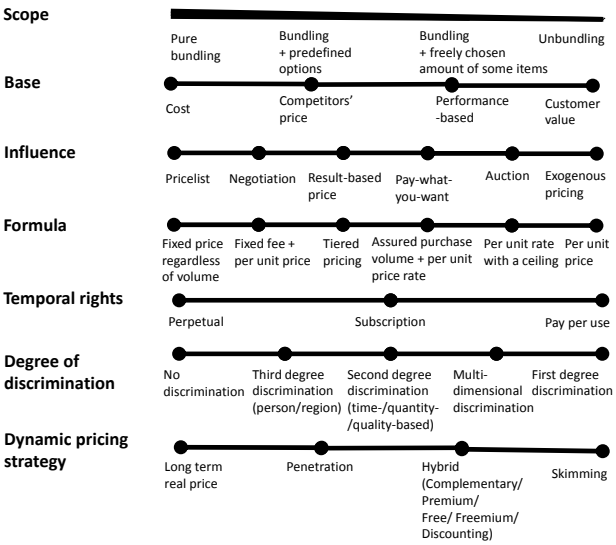

Fig. 2. Cloud Solution Pricing Framework

the cases Attribute and Package. Based on the literature, we identify the categories *Package* as *Pure bundling* and *Attribute* as *Unbundling*. The combination of these is referred in the literature to as *Customized bundling*, where customers can choose the components of the bundle while the seller determines the price and scope of the bundle [21]. In IT industry, we see examples of customized bundling when even the price and the scope of the bundle is negotiable. To ease the process of determining the scope level, we propose the categories [Bundling where the amount of some items can be chosen from predefined options] and [Bundling where the amount of some items can be chosen freely].

Tiered Pricing. We propose to add a new item to the Formula dimension for offerings with a fixed price and a limitation on the volume or the functionality, where the user has to switch to a less-limited offering with a different price if (s)he requests more volume or functionality. Named as *Tiered-pricing*, the formula attempts to package services and products by matching price levels to user's willingness-to-pay [14]. This formula is popular among IT offerings that apply vertical versioning.

Subscription-Based Pricing Models. In the Temporal rights dimension of SBIFT model the authors distinguish between *Leasing*, *Renting* and *Subscription*. However, these three concepts are faded in cloud literature (see e.g. [17,5]), therefore we propose to use the term Subscription meaning Renting and Leasing as well and leaving Renting and Leasing out of the framework as separate items.

Usage-Based Pricing Models. In cloud literature, the term Pay per use pricing is used when the customer is charged on the actual usage, that has to be

monitored and measured [22]. The customer does not have to make any commitment to use the service or product for a predefined period: there is no obligatory monthly fee, the user pays for the used volume. In digital content pricing literature, units represent a pricing metric that can be either linked to the actual usage or volume of the service/product (usage-dependent metric) or represent only the usage potential (usage-independent metric) [7,20]. Hence, the term usage-based pricing known from cloud industry refers to a SBIFT price model, where the Formula dimension is Per unit price with a usage-based metric and the Temporal rights is Pay per use.

Performance-Based Pricing. Being a broadly used pricing strategy in integrated solution pricing, we propose to add the category *Performance-based pricing* to the Base dimension, that takes into consideration both the suppliers' costs and the customers' perceived value. In this case, the seller guarantees a certain performance level for a negotiated price and pays a penalty if this is not achieved [15,23].

Proposed Dimension: Degree of Discrimination. Based on literature review and the wide use of this pricing aspect of our data sample, we propose to add the dimension Degree of discrimination to the SBIFT model. Price discrimination is used when the same product/service is offered for different buyers for different price. This strategy is extremely important for providers of digital goods, since the low marginal costs allow them to sell the offering also for customers with low willingness to pay [7]. The categories of the dimension are proposed as follows.

The left most item is *No discrimination*, meaning that the product/service is offered for the same price for everybody. In case of *First degree discrimination* the vendor offers the same product/service with different prices for different customers. *Second degree price discrimination* is used when providers sell different units of output for different prices [24]. In this case, customers use self-selection to choose from the offers [25]. Second degree price differentiations can be quantity-, time- and quality-based [7]. In case of *Quantity-based price discrimination* the price depends on the amount of the bought goods [24]. When prices differ in different points of times, *time-based price discrimination* is used. In case of *Quality-based price discrimination* different product/service variants are offered with different price [26]. When applying *Third degree price discrimination*, the vendor identifies different customer groups based on their willingness-to-pay [26]. Third degree price discrimination can be Personal (e.g. student discounts) or Regional (e.g. different prices for developing countries) [7]. *Multi-dimensional price discrimination* occurs when price differentiation is made based on more than one dimension [7].

Proposed Dimension: Dynamic Pricing Strategy. Because of its important role in cloud pricing suggested by the literature [7], we propose Dynamic Pricing Strategy to the SBIFT model. Prices set in a dynamic environment can influence the demand behavior of price sensitive customers [27]. Dynamic pricing is the strategy where prices are not fixed for a relatively long period, but the seller

dynamically changes the prices over time, based on factors such as time of sale, demand information and supply availability. Next the categories of the dimension are proposed.

The first option is the *Long-term real price* strategy, when prices are kept the same for longer periods and they are adjusted only if necessary, not as a part of a predetermined strategy. The next option is the *Penetration strategy*, when vendors use low prices for faster market-entry and then increase prices over time [28,12]. In case of *Skimming* the vendor sets high prices in the early stages of market development and then gradually reduces the prices to attract also more price sensitive market segments [12]. *Hybrid pricing strategies* [14] combine elements of penetration and skimming strategies and may contain for example: *Complementary pricing* [14], *Premium pricing* [14], *Free* [8], *Freemium/Follow-the-free* [8,7] or *Random or periodic discounting* [14].

4.2 Pricing Models in Cloud Industry

Our analysis shows, that indeed, currently used pricing models are very complex, difficult to understand and compare (in line with [8,29]). Solutions appear as a result of co-operation and competition between the actors of the ecosystem, and the interconnectivity between the actors is visible also in the pricing models (in line with [30]). In Figure 3, currently used pricing model characteristics of different service sectors are marked, where the values inside the rectangles describe the rounded usage proportions of the respective pricing aspect. In the picture the most popular pricing patterns and the most rarely used categories are also shown. Results related to the dimensions *Base* and *Dynamic pricing strategies* are missing from the figure, since there was not enough data regarding these two aspects. It can be seen from the figure, that firms use similar pricing models for IaaS, Paas and SaaS offerings.

Most Popular Pricing Model Patterns

Based on our analysis, we can conclude that cloud providers indeed differentiate by price since there is a big diversity in applied pricing models. The most popular pricing model is *[Pure bundling, Pricelist, Tiered pricing, Subscription and Second degree discrimination]* for all IaaS, PaaS and SaaS offerings, being applied in more than 20% of the cases. Price bundling is an effective pricing strategy if variable costs are near zero, or at least relatively low compared to the customers' willingness to pay. On the other hand, using different price bundling and unbundling solutions result in a nontransparent market because of the difficulties in price comparisons, and that effects negatively both the providers and the customers [29]. Pricelists are broadly used in cloud industry, especially when there is a large customer base with similar needs. In case of IaaS offerings, another popular pricing model is revealed since IaaS offerings are priced in 20% of the cases with the pricing model *[Pure bundling, Pricelist, Assured purchase volume plus per unit price, Subscription and No discrimination].* As a difference to the price model above, customers get the same product for the same price

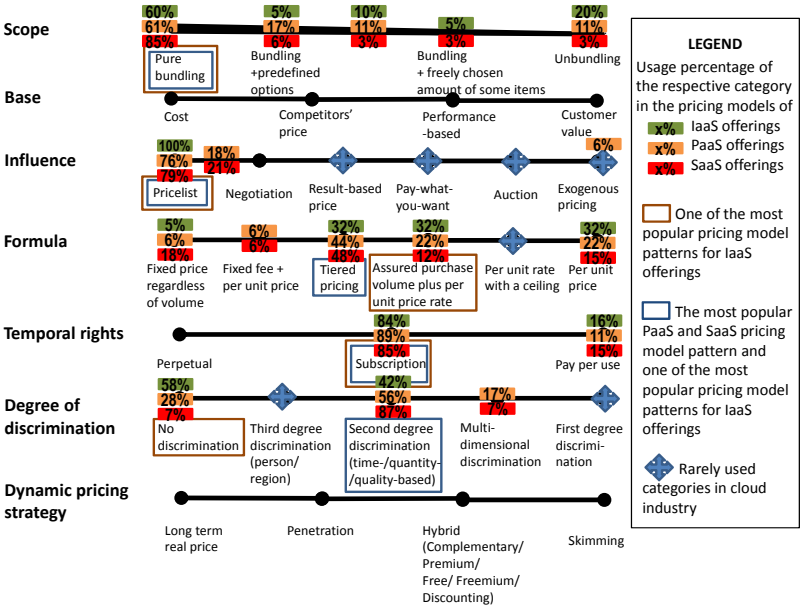

Fig. 3. Currently used pricing models in the cloud industry

without any discrimination, and they have the option to buy additional resources with a predefined unit price.

Our study revealed also, that *Free trial version* is offered to the users in 10%, 90%, and 56% of IaaS, PaaS and SaaS offerings, respectively. Besides this hybrid strategy, we met examples of *Tiered marginal discounting*, which assures that usage increase is not so painful while usage decrease still brings economic benefits for the customer.

Rarely Used Categories

Despite of the big diversity in cloud pricing, there are still rarely used categories that may provide differentiation for firms. Based on our findings, one of the rarely used categories is *Result-based pricing*. However, this category may be often used among business partners, where the actors of the value chain split the generated revenue. Examples of rarely used *Pay-what-you-want* pricing are the popular games downloadable from Humble Bundle website[2] [31]. *Auction pricing* is also rarely used, however, a good example from IaaS industry could be Amazon's pricing model regarding the EC2 Spot Instances. On the other hand, Shapiro and Varian [32] state that auctions is usually not a viable option for digital goods where the incremental cost of production is zero. Examples of *Exogenous pricing* are found -however rarely- in SaaS pricing: solutions are priced partly based on the pricing model of IaaS provider - in this case, neither the SaaS provider nor the customer have an influence on this price component.

[2] `http://www.humblebundle.com/`

No examples have been found by the authors for the use of *Per unit rate with a ceiling* in cloud industry. Our study reveals, that *Third degree discrimination* is not used alone, but it is preferred to be applied together with *Second degree discrimination*. In addition, *First degree discrimination* is rarely used in cloud context, probably because providers have difficulties in acquiring knowledge on each user's willingness-to-pay [7].

5 Conclusions and Further Research

Pricing is a strategic tool in manager's hands, where finding a good price model brings success for the companies. On the other hand, it is a challenging task with long-term consequences, where decision makers have to take into consideration many factors, such as the offering itself, the target market segment with specific customer needs, the competitors' similar offerings, the costs, etc. With the sudden growth of different cloud solutions, also pricing has become increasingly complex resulting in a "constantly changing labyrinth" of pricing [8]. In this research, we attempted to find a systematic way to describe the pricing models in order to help decision makers plan, develop and speak about pricing alternatives. The proposed 7-dimensional model is an extended and customized version of the SBIFT model developed for cloud industry, that takes into consideration both the general knowledge about pricing and the specific cloud characteristics.

In this paper, an empirical study has been carried out in order to identify the currently used pricing models of the cloud solutions. We found, that the pricing models of IaaS, PaaS and SaaS offerings have similar patterns, that leads us not to distinguish between different service categories but rather concentrate on pricing of cloud solutions. In line with Kihal et al. [29] and Cusumano [8], we found out also, that the big diversity in the pricing models makes price comparison difficult.

Our study has some limitations that provide avenues for further research. Besides our analysis of pricing information available online, data has to be gathered and studied from other sources as well, e.g. through cases studies or quantitative research. In further research, dependencies between the dimensions and categories have to be studied also. Because of the the dynamic nature of cloud value networks [33], the interaction between different actors of an ecosystem has an impact also on pricing. Offerings are interconnected and pricing models have to be established in a complex service system with multiple stake-holders [30]. Further work is needed to analyze how the pricing models of different actors enable or limit each other's pricing models [11].

References

1. Weinhardt, C., Anandasivam, A., Blau, B., Borissov, N., Meinl, T., Michalk, W., Stößer, J.: Cloud computing–a classification, business models, and research directions. Business & Information Systems Engineering 1(5), 391–399 (2009)

2. Anandasivam, A., Premm, M.: Bid price control and dynamic pricing in clouds. In: Proceedings of the European Conference on Information Systems, pp. 1–14 (2009)
3. Piercy, N.F., Cravens, D.W., Lane, N.: Thinking strategically about pricing decisions. Journal of Business Strategy 31(5), 38–48 (2010)
4. Gourville, J., Soman, D.: Pricing and the psychology of consumption. Harvard Business Review 80(9), 90–105 (2002)
5. Ojala, A.: Software renting in the era of cloud computing. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 662–669. IEEE (2012)
6. Porter, M.: Competitive strategy: Techniques for analyzing industry and competitors (1980)
7. Lehmann, S., Buxmann, P.: Pricing strategies of software vendors. Business & Information Systems Engineering 1(6), 452–462 (2009)
8. Cusumano, M.A.: The changing labyrinth of software pricing. Commun. ACM 50(7), 19–22 (2007)
9. 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)
10. Hinterhuber, A.: Towards value-based pricing: An integrative framework for decision making. Industrial Marketing Management 33(8), 765–778 (2004)
11. Iveroth, E., Westelius, A., Petri, C.J., Olve, N.G., Coster, M., Nilsson, F.: How to differentiate by price: Proposal for a five-dimensional model. European Management Journal (2012)
12. Shipley, D., Jobber, D.: Integrative pricing via the pricing wheel. Industrial Marketing Management 30(3), 301–314 (2001)
13. Danziger, S., Israeli, A., Bekerman, M.: The relative role of strategic assets in determining customer perceptions of hotel room price. International Journal of Hospitality Management 25(1), 129–145 (2006)
14. Harmon, R., Demirkan, H., Hefley, B., Auseklis, N.: Pricing strategies for information technology services: A value-based approach. In: 42nd Hawaii International Conference on System Sciences, HICSS 2009, ID: 1, pp. 1–10 (2009)
15. Bonnemeier, S., Burianek, F., Reichwald, R.: Revenue models for integrated customer solutions: Concept and organizational implementation. Journal of Revenue & Pricing Management 9(3), 228–238 (2010)
16. Sundararajan, A.: Nonlinear pricing of information goods. Management Science 50(12), 1660–1673 (2004)
17. Ojala, A.: Revenue models in saas. IT Professional (2012)
18. Ferrante, D.: Software licensing models: What's out there? IT Professional 8(6), 24–29 (2006)
19. Choudhary, V.: Comparison of software quality under perpetual licensing and software as a service. J. Manage. Inf. Syst. 24(2), 141–165 (2007)
20. 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)
21. Hitt, L., Chen, P.: Bundling with customer self-selection: A simple approach to bundling low-marginal-cost goods. Management Science, 1481–1493 (2005)
22. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM 53(4), 50–58 (2010)

23. Becker, M., Borrisov, N., Deora, V., Rana, O.F., Neumann, D.: Using k-pricing for penalty calculation in grid market. In: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, ID: 1, p. 97 (2008)
24. Varian, H.: Differential pricing and efficiency. First Monday 1(2-5) (1996)
25. Spiegel, Y.: Second degree price discrimination. Tel Aviv University (1997)
26. Varian, H.: Versioning information goods. University of California, Berkeley (1997)
27. Anandasivam, A., Buschek, S., Buyya, R.: A heuristic approach for capacity control in clouds. In: IEEE Conference on Commerce and Enterprise Computing, CEC 2009, pp. 90–97. IEEE (2009)
28. Dean, J.: Pricing pioneering products. The Journal of Industrial Economics, 165–179 (1969)
29. El Kihal, S., Schlereth, C., Skiera, B.: Price comparision for infrastructure-as-a-service
30. Ng, I.C.L.: The future of pricing and revenue models. Journal of Revenue & Pricing Management 9(3), 276–281 (2010)
31. Jurisic, M., Kermek, D.: Taxonomy of digital economy business models. In: MIPRO 2011 Proceedings of the 34th International Convention, pp. 1414–1419. IEEE (2011)
32. Shapiro, C., Varian, H.: Information rules: a strategic guide to the network economy. Harvard Business Press (1998)
33. Ojala, A., Tyrväinen, P.: Value networks in cloud computing. Journal of Business Strategy 32(6), 40–49 (2011)

The Impact of Software-as-a-Service on Software Ecosystems

Sebastian Walter Schütz[1], Thomas Kude[1], and Karl Michael Popp[2]

[1] University of Mannheim
sebastian.w.schuetz@gmail.com,
kude@uni-mannheim.de
[2] SAP AG
Walldorf, Germany
karl.michael.popp@sap.com

Abstract. The trend towards cloud-based applications changes the way customers run their businesses, but also the way software is sold and delivered. This affects software ecosystems and the way software vendors interact and manage their partners. In order to explore on the impacts, we conducted a single-case study by examining a globally leading software vendor of both, traditional on-premises software as well as cloud solutions. The study reveals new insights on how the SaaS revolution impacts partner management within software ecosystems from a vendor perspective, for instance that successful cloud partners may not necessarily come from a cloud background.

Keywords: Software-as-a-Service, Cloud, Software Ecosystem, Partner Management, Software Vendor.

1 Introduction

The emergence of Software-as-a-Service (SaaS) has brought a wind of innovation to the software industry, shifting the ownership, delivery and management of software from the customer to the vendor [12]. Transforming the traditional business models to SaaS brings a specific set of benefits to the business, especially addressing the pain points of customers: software updates, infrastructure maintenance, and capacity planning [30,24]. These activities are now entirely performed by the vendor, relieving customers from many responsibilities. At the same time, the total cost of ownership decreases [10]. However, along with these changes, new challenges arise - especially for the vendor's value propositions, revenue models, and sales channels [26]. While SaaS has initially addressed small and medium sized businesses, large accounts are becoming more and more relevant targets - creating a large potential customer base for which a vendor may not have the resources to face by itself. As a result, vendors try to engage in external partnerships, so called software ecosystems, which have become an important go-to-market strategy for SaaS vendors. Traditionally, these external firms act as resellers, complementing software developers or system integrators

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 130–140, 2013.

and therefore bring in additional resources, knowledge and customer access [1]. While these ecosystems are already existing for the traditional on-premises business, the emergence of SaaS may cause them to change [13] - making it necessary for software vendors to cautiously manage their ecosystem and thus their old and new partners [14].

Recent research in the intersecting field of SaaS-based software ecosystems has focused on the ecosystem's characteristics and implications for software vendors. However, impacts on existing on-premises software ecosystems, which are in a process of transition towards the new technology, have not been studied yet. Therefore, this single-case study, conducted within a traditional, globally leading software vendor, is intended to answer the question of how and why the emergence of SaaS affects such traditional software ecosystems as well as what possible implications the shift to SaaS may have for transitioning on-premises software vendor.

In a first step, the relevant literature will be briefly reviewed - in particular, a definition for the terms SaaS and software ecosystem will be given and the current research state presented. Furthermore, the intersecting research will be discussed and the research gap highlighted. Subsequently, the case will be introduced and the results will be presented referring to three a priori defined areas of impact. Finally, the findings are discussed and limitations and possible future work will be highlighted.

2 Literature Review

This chapter will discuss previous research on SaaS, ecosystems and on the intersection between the two areas. In doing so, the identified research gap will be highlighted.

2.1 Software-as-a-Service

Wilhite [29] recently defined SaaS as a part of the services offered within a Cloud Computing environment, referring to the hardware and software of large data centers. Accordingly, Cloud Computing comprises the services being sold from these data centers: either software (SaaS) or utilities (Platform-as-a-Service and Infrastructure-as-a-Service). Present research already covers several technical and business aspects of SaaS, such as customization [27], software architecture [25] as well as integration and software development aspects [16]. But also benefits [24], value proposition, adoption issues and risks [30] as well as a future market prognosis [19] have been subject of research already. Even though all of these studies examined the SaaS technology from different perspectives, Stuckenberg and Fielt [26] were among the first to find evidence that software ecosystems might be affected as well. Similarly, Kim and Korea [19] already predicted the emergence of cloud computing ecosystems and thus highlighted the connection to ecosystem research.

2.2 Software Ecosystems

The term software ecosystems is defined by Jansen as "a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artefacts" [18, p. 34]. Within such ecosystems, the roles of participants can be divided into keystones (platform providers) and niche players (firms who participate in the ecosystem) [15]. Building on this, Kittlaus [20] specified the role of niche players even further by introducing the types of niche players as shown in Table 1.

Table 1. Partner roles niche players can assume (based on [20])

Partner Type	Abbrev.	Definition
Value Added Reseller	VAR	Extends the direct sales channel
Independent Software Vendor	ISV	Sells complementary products
Original Equipment Manufacturer	OEM	Embeds the keystones products
System Integrator	SI	Offers implementation

The previously mentioned research targeted software ecosystems in general. While research on how SaaS as one specific platform affects the surrounding ecosystem - especially on ecosystems underlying a transformation towards this new concept - is still limited, some recent studies have started to address questions at the intersection of SaaS and software ecosystems. The current state of research will be presented in the following.

2.3 SaaS in Software Ecosystems

From a conceptual perspective, Cusumano [7] examined whether the SaaS model has the capabilities to serve as a technology platform around which a software ecosystem can emerge. Along four levers for generating networks effects and building ecosystems in general, Cusumano attests SaaS as well as the more general Cloud model the potential to be come a new ecosystem platform. However, Cusumano [8] particularly highlights the increased possibility of conflicting interests when software vendors opening up their platforms for complementors. Similarly, Stuckenberg and Fielt [26] found evidence that internal conflict within ecosystems may increase as a result of the reduced customization possibilities of SaaS solutions and thus reduced differentiation between competitors.

Looking at SaaS in software ecosystems from an empirical perspective, Hilkert et al. [14] examined where exactly SaaS drives changes within the software industry. The goal was to identify areas of change regarding characteristics specific to the IT industry, using a transaction cost based approach to analyze two major

software vendors: salesfore.com as a pure SaaS vendor, and CAS genesisWorld as a pure on-premises vendor. As a result of this explanatory study, their findings comprise four major areas. First, it was predicted that SaaS will lower the total costs of an entire solution as well as raise the variety of complementary offerings for the customers. Second, for intermediaries, the change may shift their role to more trust and relationship building activities. Third, for complementors, the competition may raise for established ISVs as entry barriers fall. And at last, software vendors will face new challenges regarding the shift from managing single partner relationships towards a market-organized ecosystem.

As noticed in Section 2.1, research on SaaS has mainly focused on business and technical aspects of both the customer as well as of the vendor side. Within in software ecosystem research, the impact of a technology platform such as SaaS remains mostly unexplored. As a notable exception, Cusumano [7] sheds light on this area from a conceptual view. Adding to these findings, this study is among the first to empirically study ecosystems in a software service environment. As opposed to Hilkert et al. [14] who focused on pure SaaS and pure on-premises vendors, traditional on-premises software vendor that enter the SaaS market and thus transition its ecosystem is not explored yet. The goal of this study is to fill this gap.

Summarizing the existing literature, the effects of SaaS on software ecosystems relate to three major areas. Hilkert et al. [14] suggest that firstly, SaaS will change the roles of partners or the *partner profiles* and secondly also the *relationship* between platform vendors and partners. Third, changes may be observed on an *ecosystem level* and relate to the competition among organizational actors [8,7,26]. Hence, implications of the shift towards SaaS can be examined on an organizational, inter-organizational, and ecosystem level. These areas derived from the review of existing literature will be used to guide our data collection and analysis.

3 Case Study

This section will present the case study's empirical results. In particular, we first elaborate on the chosen research design and the data collection. Then, detailed information on the data sources are given and reasons why they have been chosen are discussed, followed by an introduction of the method of analysis. The gathered evidence will then be categorized along the sub-dimensions partner profiles, relationships, and ecosystem. Then, the results will be discussed.

3.1 Research Design

According to the open research question, an exploratory single-case study was conducted with the goal to improve our understanding of the implications of SaaS on well-established software ecosystems. Thus, the subject is a globally leading business software vendor for traditional on-premises solutions, which recently entered the SaaS market and maintains an extensive software ecosystem.

As a rationale for the single-case research strategy, the revelatory opportunity resulting from a unique access to the case site was leveraged [31,11].

The data collection consisted of three semi-structured interviews which have been conducted with key personal of the case organization. Within a semi-structured interview, the interviewer sets up a predefined structure prior to the interview and then follows this structure while still being flexible enough to adjust to new topics and discussions. It is thus suitable for conducting interviews with a limited number of people, for which the mentioned flexibility as well as the appreciation of context to understand the individuals perception may result in the generation of rich data, as it is done within this study. The detailed sequence of the interview formed during the interview process [6,9]. As suggested by [31], the three areas introduced in Section 2.3 guided and structured the process of data collection and the analysis of the empirical evidence. These areas are generally considered abstract enough to capture our empirical insights. Accordingly, the following three topics are covered during the interviews:

1. *Partner profiles:* How does SaaS partners differ from on-premise partner? What characteristics and assets do partners need to possess in order to succeed in a SaaS-based ecosystem? What entry barriers do they encounter?
2. *Relationship:* How does SaaS affect the relationship to partners, especially the fact that systems are not longer accessible on-site?
3. *Ecosystem level:* Does the standardization of products encourage competition? If yes, how can possible conflicts be avoided or resolved?

3.2 Data Sources

For this exploratory study, the data sources comprised three individuals within the case subject's organization. As a result of their roles within the case organization, they possess unique knowledge and experience about their field as well as the organization.

Individual A: The first interview was conducted with a global cloud channel development manager, who is responsible for partner profiling, recruitment and channel cloud business models. The individual draws on five years of cloud channel experience in total starting from 2007 (which is the same year the organization's first cloud solution was offered), whereof three years in a local role at the company's headquarters and two years in the current global role. As a result, the individual is able to provide valuable insights with regards to development, challenges and particular necessary changes to make the organization's cloud channel run successfully.

Individual B: The second interview partner has been responsible since four years for the partner management within the cloud channel, especially in Central Europe. The individual B has been part of the organization since twelve years and was working - prior to his partner management role - in the strategy department, where s/he was involved in the planning of the offered cloud service, especially design of sales cycles, partner models and the go-to-market approach. Thus, the individual can contribute with insights from a strategy perspective, in particular

on how partners approach the cloud topic and how the globally leading software vendors manages its partnerships.

Individual C: The third individual's role is responsible for partner recruitment in EMEA (Europe, Middle East and Africa). The individual joined the organization almost ten years ago and has been working in a channel role since five years. According to his role, the individual contributes through insights into the ecosystem building strategies and activities and is particularly able to share insights on cloud partner profiles and types.

3.3 Data Analysis

The goal of this study is to provide a better understanding of the implications of the shift towards SaaS for well-established platform ecosystems. More specifically, after having deduced three potential areas of impact from extant literature - partnership profiles, relationships, and ecosystem - the goal of our empirical study was to fill these areas with life in the context of a large and globally acting enterprise software vendor. In order to do so, we carefully analyzed the collected qualitative data and assigned text fragments to the three identified areas [31]. Even though we were open for new areas of impact to emerge, all our findings could be subsumed under the three a priori identified categories. In an iterative process of sensemaking, we reexamined the coded text fragments and structured them according to several distinct patterns that are presented subsequently (see also Table 2).

3.4 Empirical Results

In this section the empirical results will be presented. Similar to the data collection, the results will be presented along the three areas of impact identified in Section 2.3: the first subsection will discuss the evidence gathered about a possible impact of SaaS on partner profiles. Then, in the second subsection, the relationship between platform vendor and partner will be elaborated, followed lastly by the impact on ecosystem level.

Partner Profiles. During the interviews it was found that in contrast to traditional on-premises software ecosystems, hybrid ecosystems will shift from a partner landscape dominated by Value Added Reseller towards a landscape that will mostly consist of Independent Software Vendors, Original Equipment Manufacturers as well as Business Process Outsources. This is caused by the nature of the platform strategy, which focuses on implementing only around 80% of the product's functionality and specifically creating business opportunities (niches) for its partners. As agreed by Cusumano [8], this is an important lever for the ecosystem's success and allows ISVs to build solutions to fill these niches. The VAR partner type, which is strong regarding marketing and sales, will no longer be able to simply resell. In consequence, this means for hybrid ecosystems that the share of value creating partners will raise compared to reselling partner types.

"Basically you use [ISVs] for both: to extend the solution, but also as a strategic factor with regards to specialization and to fill segments we do not want to address ourselves. (Individual A)"

Additionally, one more partner was suggested to play an increasingly important role in future SaaS ecosystems: the business process outsourcer (BPO) [22]. With IT being outsourced, hosted, owned, and delivered by a third party, it is only standing to reason that these service providers also take over the responsibility of processes, e.g. payroll. Regarding the requirements for a partner to be successful, it is suggested that the most important assets are market access and expertise in the specific domain - other than cloud. This is necessary since customers do not buy cloud solutions, but a solution addressing a certain area - e.g. sales.

"We used to have partners who came from a Google Cloud background and it happened that those struggled with their approach since they didn't have any idea about ERP. (Individual A)"

Another requirement for partners can be derived from the positive adoption factor articulated by Xin & Levina [30], arguing that customer's high costs of capital positively affects the decision towards a SaaS adoption. Due to the SaaS business model, a partner's cash flow is split into monthly fees over a period of time, resulting in the situation in which the initial customer acquisition costs cannot be amortized immediately and thus requires strong financial liquidity on the partner side. However, it is further argued that an investment is always necessary to enter a new field of business - no matter whether on-premises or cloud. To highlight the difference in necessary investment between a cloud business and on-premises business, the number of sales employees recommended for starting a partnership can be compared. While for traditional business partnerships (e.g. VAR) a hybrid sales person is suggested to suffice, for SaaS at least three full-time sales employees are recommended, due to shorter sales-cycles and smaller deal sizes. Further, since cloud is a new business, a hybrid person would rather try to sell the well-known product instead of putting effort into positioning the cloud solution.

Relationship. During the interviews it is stated that the current, project-based business of System Integrators may no longer meet market demands. According to the interviewee's perception, the software market demands packaged solutions delivered as a service in such a way that customers do not longer carry risks of implementation projects by themselves - but rather engage in a long-term relationship with the software supplier. This is in line with Hilkert et al. who found a shift towards deeper relationships and an increasing importance of trust as an essential factor leading to successful partnerships [14].

"Nowadays, a SI's business is to enter the customer, implement a project and then exit. This is the way a [SI] partner runs business today but also a serious problem. (Individual B)"

Ecosystem. Previous research suggested a more intensive competition within software ecosystems resulting from the shift towards SaaS [26,8]. However, our empirical data suggests that the relatively young SaaS business currently offers plenty of opportunities and unoccupied niches. Hence, competition is not expected to be fierce from a short term perspective. Yet, in the long run, competition is expected to increase, causing the differentiation to take place rather on a service level and company branding than on product features.

> *"... I'm confident that quality will win. Which means that partners who enter the market today - and earlier - will gain experience, develop add-ons and build industry-specific customizations [...] and thus will again be able to differentiate towards certain customers and therefore won't feel competition too much. (Individual A)"*

Accordingly, it is suggested that partners who enter the market early and build up specialized, high-quality domain knowledge will win over the competition similar to the traditional business. By contrast, partners that fail to differentiate and chose a generalist strategy may struggle.

Within SaaS ecosystems, responsibilities for the software as well as the delivery mostly shift towards the platform vendor (e.g. hosting and maintenance is done in the vendor's data center). While this may increase the power imbalance between platform vendors and partners and may therefore be perceived as a potential threat, our data suggests that, again due to the rather new SaaS business, competing with partners in a certain niche may not be in the interest of platform vendors.

> *"If a partner creates a niche solution, we are receiving 30% for no effort. If we do it ourselves, we are receiving 70% for 100% effort. This is a simple business calculation." (Individual B)*

Table 2. Possible impacts of SaaS on software ecosystems

Area	Findings
Partner Profiles	- Shift from VAR partners towards ISVs, OEMs, BPOs - Financial liquidity, marketing & domain expertise is required
Relationship	- SI's business model may no longer meet market demands
Ecosystem	- Short term: no high partner vs. partner competition - Early adopters may take the lead similar to the on-promises business - Short term: more advantageous position of keystone may not be a threat

4 Discussion and Conclusion

This study contributes to recent research by exploring the impact of the emergence of SaaS on existing on-premises software vendors. Whereas previous research explored possible impacts on newly formed SaaS-only vendors and their ecosystems [14], the scenario of traditional ecosystems which transition into SaaS ecosystems has not been covered yet. This may be of particular interest, given

that the majority of participants in a SaaS-based ecosystem can be assumed to come from an on-premises background. This study is among the first to empirically analyze impacts of SaaS on these software ecosystems that originate in an on-premises environment.

Our empirical anlysis provides several insights. Firstly, for the *partner profiles*, the major partner type for SaaS ecosystems may be expected to shift towards ISVs, OEMs and the new BPO partner type. Furthermore, these partners operating within a SaaS ecosystem may face changed market requirements, especially regarding financial liquidity, marketing and domain expertise. In contrast to the intuitive expectation that partners with a Cloud or SaaS background may succeed in SaaS ecosystems, our findings suggest that traditional on-premises partners with a strong niche focus may posses a substantial advantage. As a consequence, transitional ecosystems may likely be dominated by traditional on-premises partners. For the area *relationships*, it can be noted that the business models of SIs may undergo a change away from project-based implementation business, also caused by the reduced customization capabilities as described by Stuckenberg and Fielt [26]. For instance, following Hilkert et al. [14], SIs may consult existing customers with security problems related to SaaS. As for the area *ecosystem*, it was found that in contrast to the predictions of existing studies on pure SaaS ecosystems, a more dynamic point of view should be taken: as the market of SaaS is still very young, the level of competition may not increase for transitional ecosystems in the short-term. However, in the long run, firms may compete on a service level, as products are heavily standardized and hence leaving little room for product differentiation at the partner side.

Our findings have several implications for the transitional software vendors. First, it is important to drive niche creation within the ecosystem in order to forward market demands to partners and create business opportunities. Furthermore, this will also reduce competition and make the overall ecosystem more attractive. Secondly, a platform vendor who extends its ecosystem towards SaaS technology may rather focus on enabling existing on-premises partners as opposed to recruiting new SaaS-only partners. This is because existing domain experts who specialize on certain niches (similar to their on-premises expertise) are suggested to succeed. In regards to new partners, the strategy should embrace ISVs, OEMs and BPOs - the latter being usually not in the focus of traditional ecosystems. Third, by modelling the go-to-market strategy, platform vendors are advised to create differentiation possibilities on a service level, as this will define competition between partners which target the same niches.

This study focused on the impact on partner management in general. By contrast, technical details in relation to partner management have not been in the main focus of this study. Thus, future work could include research on impacts of SaaS on integration aspects between complementors such as between two businesses or different SaaS hosts. Furthermore, the empirical evidence gathered within this study only comprises one organization. Accordingly, deeper insights may be gained by replicating this study within multiple cases in the future. Likewise, during the interviews, the impact of legal aspects between service offering

and product offering in relation to partnerships have not been covered. Therefore, further interviews with legal experts may add valuable insights to this as well.

References

1. Bosch, J.: From Software Product Lines to Software Ecosystems. In: International Software Product Line Conference, pp. 1–10 (2009)
2. Campbell, P.R.J., Ain, A., Ahmed, F.: A Three-Dimensional View of Software Ecosystems. In: ECSA, pp. 81–84 (2010)
3. Ceccagnoli, M., Forman, C.: When Do ISVs Join a Platform Ecosystem? Evidence from the Enterprise Software Industry (2009)
4. Ceccagnoli, M., Forman, C.: Cocreation of Value in A Platform Ecosystem: The Case of Enterprise Software. MIS Quartely 36(1), 263–290 (2012)
5. Choudhary, V.: Software as a Service: Implications for Investment in Software Development The Paul Merage School of Business. In: Hawaii International Conference in System Sciences, pp. 1–10 (2007)
6. Cohen, D., Crabtree, B.: Semi-structured interviews. Qualitative Research Guidelines Project (2006)
7. Cusumano, M.: Will saas and cloud computing become a new industry platform? In: Software-as-a-Service, pp. 3–13 (2010)
8. Cusumano, M.: Cloud computing and SaaS as new computing platforms. Communications of the ACM 53(4), 27 (2010)
9. Drever, E.: Using Semi-Structured Interviews in Small-Scale Research. A Teacher's Guide. Scottish Council for Research in Education, Edinburgh (2003)
10. Dubey, A., Wagle, D.: Delivering software as a service. The McKinsey Quartely (2007)
11. Eisenhardt, M.: Building Theories from Case Study Research. Academy of Management Review 14(4), 532–550 (1989)
12. Gartner: Software As A Service (2012), `http://www.gartner.com/it-glossary/software-as-a-service-saas/`
13. Knight, C., Munro, M., Gold, N., Mohan, A.: Understanding service-oriented software. IEEE Software, 71–77 (2004)
14. Hilkert, D., Wolf, C.M., Benlian, A., Hess, T.: The "As-a-service"-paradigm and its implications for the software industry – insights from a comparative case study in CRM software ecosystems. In: Tyrväinen, P., Jansen, S., Cusumano, M.A. (eds.) ICSOB 2010. LNBIP, vol. 51, pp. 125–137. Springer, Heidelberg (2010)
15. Iansiti, M., Levien, R.: Strategy as Ecology. Havard Business Review (2004)
16. Integrating legacy Software into a Service oriented Architecture. In: Proceedings of the Conference on Software Maintenance and Reengineering, no. 3-14. IEEE Computer Society (2006)
17. Jansen, S., Brinkkemper, S., Finkelstein, A.: Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems. In: International Workshop on Software Ecosystems, no. 2, pp. 34–48 (2009)
18. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: 2009 31st International Conference on Software Engineering - Companion Volume, pp. 187–190 (2009)
19. Kim, W., Korea, S.: Cloud Computing: Today and Tomorrow. Journal of Object Technology 8(1), 65–72 (2009)

20. Kittlaus, H., Clough, P.: Software Product Management and Pricing: Key Success Factors for Software Organizations. Springer (2009)
21. Kude, T., Dibbern, J., Heinzl, A.: Why Do Complementors Participate? An Analysis of Partnership Networks in the Enterprise Software Industry. IEEE Transactions on Engineering Management 59(2), 250–265 (2012)
22. Lacity, M., Solomon, S., Yan, A., Willcocks, L.: Business process outsourcing studies: a critical review and research directions. Journal of information technology 26(4), 221–258 (2011)
23. Mohr, J., Spekman, R.: Charactersitics of partnership success: Partnership attributes, communication behavior, and conflict resolution techniques. Strategic Management Journal 15, 135–152 (1994)
24. Nordström, H.: Evaluating the Software as a Service Business Model: From CPU Time-Sharing to Online Innovation Sharing. In: IADIS International Conference e-Society, no. 2000, pp. 177–186 (2005)
25. Papazoglou, M.P.: Service -Oriented Computing: Concepts, Characteristics and Directions. In: International Conference on Web Information Systems Engineering (2003)
26. Stuckenberg, S., Fielt, E.: The Impact of Software-as-a-Service on Business Models of Leading Software Vendors: Experiences From Three Exploratory Case Studies. Association for Information Systems (2011)
27. Sun, W., Zhang, X., Guo, C.J., Sun, P., Su, H.: Software as a Service: Configuration and Customization Perspectives. In: 2008 IEEE Congress on Services Part II (services-2 2008), pp. 18–25 (2008)
28. Tuten, T.L., Urban, D.J.: An Expanded Model of Partnership Formation and Success. Industrial Marketing Management 164, 149–164 (2001)
29. Wilhite, S.E.: A View of Cloud Computing. HDA now / Hawaii Dental Association, 12 (2012)
30. Xin, M., Levina, N.: Software-as-a-Service Model: Elaborating Client-side Adoption Factors. In: The 29th International Conference on Information Systems, pp. 1–12 (2008)
31. Yin, R.K.: Case Study Research: Design and Methods, 3rd edn. SAGE Publications (2003)

Towards a Conceptual Framework for Assessing the Benefits of Cloud Computing

Nattakarn Phaphoom, Xiaofeng Wang, and Pekka Abrahamsson

Faculty of Computer Science
Free University of Bolzano
Piazza Domenicani 3 Bolzano 39100 Italy
phaphoom@inf.unibz.it,
{xiaofeng.wang,pekka.abrahamsson}@unibz.it

Abstract. The understanding of cloud computing's benefits is fraught with misconception. Prospect adopters often underestimate, overestimate, or do not thoroughly consider the benefits from all relevant perspectives. This is mainly due to a lack of appropriate benefit identification and assessment mechanisms for this specific technology. In addition, the benefits reported in the literatures have been scattered and unorganized. This paper is designed to identify the perspectives necessary to capture the benefits of cloud computing. A conceptual framework of cloud computing benefits is proposed, based on various benefit taxonomies presented in the Information System literature. The conceptual framework accounts for the different business areas and organizational levels where each of the benefits manifests.

Keywords: Cloud computing, benefits, benefit framework, benefit assessment, balanced scorecard.

1 Introduction

Cloud computing has brought the benefits of utility computing into a global scale. It is perceived as a shift in computing paradigms, representing a fundamental change in the way IT services are developed, offered, acquired, maintained and paid for [12,34]. In this paradigm, cloud service providers manage a pool of computing resources, generally by means of virtualization, and offer services in terms of infrastructure (IaaS), platform (PaaS) and software (SaaS) [13]. Consumers can acquire such computing services on-demand over the Internet through self-service interfaces. The quality of services is maintained according to service level agreement. Service usage is automatically metered, allowing consumers to pay only for the services that they use.

Cloud computing has promised tremendous benefits to enterprise IT in terms of cost efficiency [23,25,34,37,38], operational excellence [25,34-36,38,41,42] and innovation [30,34,36,39,42]. For this reason, organizations have started to utilize cloud-based services or seriously consider adopting these paradigms. According to Gartner [14], private cloud is among the highest interests for enterprises in 2012. As a

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 141–152, 2013.

consequence, it is foreseen that virtualization will reach mainstream adoption within two years, followed by the mainstream adoption of IaaS and PaaS within two to five years.

So far the drivers of cloud adoption have been predominantly from a cost perspective [34]. The main advantages include the ability of turning substantial upfront IT investments into operational expenses [25,34,41], the ability to lower IT management, operational and maintenance cost [23,25,37,42] and reduced datacenter's cost variability to prepare for unpredictable and changing demands [23,38]. Public cloud is of particular interest for start-ups and small and medium enterprises due to their limited investment capability and relaxed conditions for security and data privacy [15,36]. Instead, private cloud becomes an option for corporate customers with specific concerns [17,34].

It is observed that the adoption drivers have shifted from cost to other benefit areas. A recent survey performed by Dimensional Research [16] identifies satisfaction of compliance requirements, better value and competitive advantage as the top motivators for moving to the cloud.

However, several issues remain unclear, which include what benefits are to be expected in which business areas, how the benefits manifest at different organizational levels and how they contribute to the achievement of intangible strategic-level benefits. The identification and assessment of cloud computing's benefits in an adopting organization could not be precise without considering multi-dimensional nature of them.

To capture a comprehensive view of cloud computing's benefits, there is a need to identify the dimensions of benefits that cloud computing can provide to organizations. The objective of our study is to build a conceptual framework to capture the benefits of cloud computing, drawing upon various frameworks and benefit taxonomies proposed in the Information System (IS) literature. The research question that guides our study has been formulated as *what are the perspectives that organizations should consider at different organizational levels for assessing the benefits of cloud computing adoption?*

The reminder of the paper is organized as follows. In the next section, the existing work on cloud computing benefit and value analysis is summarized, followed by a review of benefit taxonomies proposed in the IS literature. The study design is explained in Section 3, after which the resulting conceptual framework for cloud benefit assessment is presented. The paper is concluded with the framework usage and the outline of the future work.

2 Related Work

2.1 Analysis of Cloud Computing Benefits

Among a large number of papers discussing the benefits of cloud computing, only a few present the benefits in an organized manner. Khajeh-Hosseini *et al.* propose a benefit assessment framework for public IaaS clouds based on 19 benefits identified from the literature [25]. In this work *benefit* is defined as "*an advantage to the*

enterprise over its status quo provided by using public IaaS clouds" and are classified as technical, financial or organizational. Mladenow *et al.* demonstrate the economic benefits of cloud computing for start-ups and SMEs through a view of value creation [36]. They propose a model that connects the effects of cloud services with four value creation drivers, namely efficiency, complementarities, networked collaboration and novelty. Phaphoom *et al*. perform a thematic analysis of a cloud computing forum to examine the perceived benefits of cloud computing for IT and software development [18]. The themes of benefits emerged from the discussion threads conclude into five areas, including availability, scalability, security, computing platform and cost & IT department. Deed and Cragg analyze the business impacts of SaaS in organizations using a generic framework for assessing IS/IT investment [19]. Four types of benefits are identified including strategic, managerial, operational and functional and support.

Existing work on cloud computing benefits appears to have several limitations. First of all, the presentation of benefits is scattered, unorganized or incomplete, making it difficult to grasp a holistic understanding. Secondly, the benefits are mostly discussed from the technical perspective, which does not sufficiently reflect the impacts on organizations. Finally, in the cases that strategic-level benefits are claimed, often there are no clear chains of evidences of how the benefits can be achieved.

2.2 Perspectives for Benefit Assessment

Benefit management literatures offer a range of appraisal techniques designed to help organizations to realize benefits from an IT investment [1,2,3]. One of the common techniques is to quantify the amount and timing of benefits of IT implementation in financial terms such as return on investment, internal rate of return and net present value [3]. Certain types of benefits require time to manifest and cannot be precisely quantified in mathematical terms. In comparison with financial evaluation, benefit assessment is relatively more complex and difficult to perform [3, 4].

Frameworks have been developed to facilitate benefit assessment. The classic framework developed by Anthony classifies benefits as operational, managerial and strategic, based on the levels of management involved [5]. Many IS benefits have been organized around this framework [6]. Shang *et al*. combine the Anthony's framework with an IT infrastructure and organizational perspective [6]. Schultheis & Summer [7], Farbey *et al*. [8] and Irani & Love [4] use an adapted version of the Anthony's framework and classify IS benefits as operational, tactical and strategic. The Balanced Scorecard approach developed by Kaplan & Norton has been used in assessing IS benefits from four perspectives including financial, customer, business process and innovation [9]. Piotrowicz & Irani combine several benefit taxonomies. In their work, the Balance Scorecard is used to identify the benefit areas and the adapted Anthony's framework is used to identify the benefit importance [10]. They further identify benefit characteristics as intangible, tangible financial and tangible non-financial. The combinations allow for a more holistic view of benefits in terms of their levels and impact areas.

In this work, the Balance Scorecard and Anthony's framework are applied to better understand the multi-dimensional nature of cloud benefits. The Balanced Scorecard is designed to provide top managers with a fast and comprehensive view of the business [9]. As thus, it identifies four areas of concerns in organizational contexts. *The customer perspective* focuses on an area of customers' concerns that generally fall into time, quality, service and cost. *The process perspective* focuses on internal operations that enable organizations to satisfy customers' needs. *The innovation and learning perspective* reflects the factors that are important for competitive success, allowing organizations to penetrate new markets and to increase their revenues. *The financial perspective* focuses on profitability, growth and shareholder values.

The Anthony's framework is applied to identify an organizational level that the benefits occur [5]. An *operational* level is concerned with ensuring that day-to-day activities are effectively and efficiently carried out. The activities are generally repeated periodically. IT is applied to streamline the process and to facilitate or replace basic, repetitive operations. *Managerial* activities are concerned with planning, controlling and monitoring the usage of organizational resources in order to support business strategic decisions. *Strategic* activities involve setting long-term goals and identifying the paths and means of achieving defined objectives. Benefits at this level are related to an achievement of organizational objectives.

3 Research Design

The main objective of our study is to identify the perspectives, in the format of a conceptual framework, necessary to capture the benefits of cloud computing in organizations. The research process involves the three steps as illustrated in Fig 1.

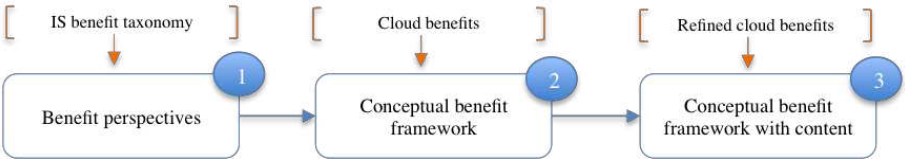

Fig. 1. The process for building the benefit framework

Step 1: Select Relevant Benefit Perspectives from IS Literature. An IS benefit taxonomy is designed to guide benefit identification and assessment in an organization. Based on the existing taxonomies presented in the IS literature, we first identified the perspectives that serve our goal. An initial benefit framework is two-dimensional, combining the Balanced Scorecard perspectives (customer, business process, innovation and financial) with the Anthony's organizational levels (operational, managerial and strategic). In this way, the framework captures both the areas of an organization's interests and the degree of impact each of the benefits introduces. In addition, both frameworks have been widely applied for benefit assessment in organizations, showing relevance and usefulness in this context.

Step 2: Improve the Framework Based on a Literature Survey. In this step we performed a literature survey to investigate whether the proposed perspectives can sufficiently capture cloud computing's benefits in organizations. Benefits stated in the cloud computing literature were used as input to improve the framework.

We searched for the studies discussing the benefits of cloud computing in the IEEE Xplore by applying the following search string on the abstracts: "cloud computing" AND (benefit OR value OR opportunity OR advantage) AND (company OR organization). The search returned 200 matching articles. After abstract reading, 18 studies were identified as relevant. Another five studies were included through a backward reference search. 148 benefits as reported in the literature were identified through a content analysis on the resulting 23 papers and categorized according to the initial framework.

The categorization allows for the discovery of new perspectives. We followed three sub-steps. Firstly, all the benefits were classified under one of the Balanced Scorecard perspectives, or as 'unclassified' if the content did not match any predefined perspectives. Secondly, the unclassified benefits were analyzed to explore new themes that emerged. New perspectives were added accordingly. Finally, the benefits belonging to each perspective were classified as operational, managerial or strategic.

The three sub-steps led to adding *IT infrastructure & services* as a new perspective along with the Balanced Scorecard. IT infrastructure benefits have been mentioned in a large number of studies [6]. IT infrastructure & services benefits are related to sharable and reusable IT resources [6, 20] and a set of related functionalities made available through IT system [11] to support one or more business areas.

Step 3: Produce a Refined List of Benefits for Each Benefit Category. Three sub-steps were performed to ensure proper classification and consistent granularity of the benefit items. First of all, the benefits were organized and refined. It included merging relevant trivial benefits into a concrete benefit item or splitting a broad benefit into a set of concrete ones. In addition to preserving granularity, this process enabled us to observe the themes that emerged for each category. Secondly, the benefit classification was verified according to the perspectives' definition. Certain benefits were reassigned to a different organizational level. Finally, a parsimony principle was applied to ensure that a minimum set of benefits were presented for each category. The resulting framework is presented in the next section.

4 A Conceptual Framework for Cloud Benefit Assessment

An execution of methodology presented in Section 3 results in the cloud computing benefit framework. The two-dimensional framework presented in Table 1 combines extended Balance Scorecard perspectives (vertical) with three impact levels of benefits (horizontal). The themes of benefits that emerged during the analysis are highlighted with bold italic characters. The detailed insights for each of the benefits are presented in this section.

Table 1. A benefit framework for cloud computing

	IT infra & service	Business process	Financial	Innovation	Customers
Strategic	- Support for green org.	-Increased business agility	- Increased investment flexibility - Ability to transform capital expense to operational expense - Removal of upfront investment	- Increased competitive advantageous - Enhanced variability & capability of products - Improved org. image	- Improved customer supports - Improved quality of provided IT services
Managerial	- Stay align with technological advancement - Simplified resource management	- Ability to run business without technical employment - Improved decision-making - Support for standardization and Streamline multiple business processes - Better consumer relationship - Improved staff productivity, leading to better time-to-market	- Removal of a long-term purchase commitment - Reduced cost of IT management - Removed/reduced real estate cost	- Extended pool of potential human resources - Extended IT resources for innovative creation - Extended collaborative network	
Operational	***IT processes & personnel*** - Removal of resource limitation problem - Removed / reduced manual maintenance work, leading to better work satisfaction - Simplified IT maintenance ***System quality*** - Support for security - Support for resiliency - Support for scalability ***Hardware & virtualization*** - Increased computing - Reduced hardware - Higher utilization - Increased flexibility in IT resource usage - Enhanced standardization of IT resources - Reduced energy consumption - Reduced carbon emission	***Process execution*** - Faster provision of IT resources to support business units - Faster business process execution ***Information*** - Ease of access to information - Reduced information latency ***Communication*** - Support for internal collaboration - Support for data sharing and exchanging with external org. - Support for consumer engagement ***Support for software development*** - Support for reusability of platform components - Support collaboration through integrated development environment - Improved application development and build processes	- Reduced IT resource maintenance cost - Reduced business operational cost - Energy cost saving	- Ability to focus on value-oriented work - Support for collective mind-share and development efforts - Support for technology integration - Expanded product channels - Enhanced analytical capability	

4.1 IT Infrastructure and Services

An Operational Level. Benefits from cloud computing appear to be associated with three areas including hardware, system quality and maintenance, IT processes and personnel. In terms of hardware, most of the identified benefits arise from virtualization. It provides a way to consolidate distributed hardware resources, allowing for standardized and centralized management [21]. Workloads from underutilized resources can be more flexibly allocated and de-allocated to serve different departments [32], leading to significant reduction in the number of servers that an organization has to maintain [39]. Public cloud takes a role as a source for on-demand resources [38,42]. Better hardware utilization, energy saving [23,25,42] and reduction of carbon emission [27,35] can be expected from a virtualized datacentre.

In terms of system quality, cloud facilitates security, resilience and scalability management processes. Supports for security arise from economies of scale and concentration of hardware resources. Security measures such as physical parameterizations are cheaper (per resource unit) when implemented at a larger scale [28,40]. It is also easier to ensure that security processes are executed in a consistent manner. Consumers of public cloud gain additional advantages from provider's investment and expertise on security. Several features such as multi-location deployment and redundancy enhance availability of systems running on cloud infrastructure [26]. Business continuity planning further benefits from geographically dispersed datacenters [26,39]. In terms of scalability, on-demand acquisition and release of resources in the cloud service model help fil the gap between demand and supply [33,34,36]. This property is useful when demand is unknown in advanced or is highly fluctuated [23].

Cloud provides a way to increase efficiency of maintenance and recovery processes. Virtualization management tools are featured with workload migration between hardware platforms, which is helpful during system upgrades [24]. To support environment preparation, virtual machine (VM) images and software modules used can be pre-hardened and patched with latest updates and security settings [40]. This process removes the manual work of environment setup, as the baseline images can be cloned and then be ready to use. Forensic images of VMs are also accessible without taking the system offline for digital investigation of failures or security breaches [40].

Changes in IT infrastructure and automation brought by virtualization improve the way IT personnel work. Manual activities such as system setup and upgrade, and failure recovery can be performed in a more automated manner. The removal of such manual work helps improve status quo of IT personnel, open a room for creative tasks and increase their work satisfaction [25, 42]. Public cloud becomes a viable option for an organization with limited IT personnel and resources [42].

A Managerial Level. Managerial benefits appear to reflect changes at the operational level. The most prevalent benefit is simplified resource management [23,24,26], due to the reduced risk of over-provisioning and under-provisioning of resources. This means a more relaxed constraint on early estimation of resource requirements [24]. Organizations using public cloud are ensured to align with technological advancements due to high competition in a cloud computing market [25].

A Strategic Level. Supports for green organizations are considered as a strategic benefit. This is due to the fact that the concept of 'green' presents a long-term initiative and captures many critical benefit areas, including efficiency of resource usage, reduction of waste, increase of utilization and energy saving.

4.2 Business Process

An Operational Level. Four themes have emerged in this category, including process execution, information, communication, and support for software development. The last theme is classified as process-related, rather than IT services-related, due to the nature of business process nowadays that is highly software intensive.

Cloud facilitates an execution of business process by speeding up computation of batch analytics [23] and transactions processing [36]. It also provides a mechanism to fasten resource provision process, allowing almost immediate access to hardware. The more efficient use of resources and time can also be expected [35].

In terms of information, cloud provides employees with flexibility and ease of access to business information from anywhere by using any standard device [28,38]. Information latency is reduced or eliminated, leading to better planning and increased efficiency of business processes [41].

In many cases, clouds are adopted to improve collaboration and communication between stakeholders [35,38,42]. Cloud platforms help ensuring quick and consistent information delivery. Web technology and centralization governed by cloud platforms facilitate collaboration among employees. Exchanging data with externals can be done more effectively. Web advancement also supports customer engagement.

To support in-house software development, cloud platforms provide developers with integrated development and deployment environments [38], allowing them to easily share their work and gain access to applications built by others. Deployment can be done rapidly on shared infrastructure. Virtualization supports reusability of platform components and it scales as users increase. [43]. When public PaaS is used, organizations can bypass problems related to software purchase, maintenance and integration. Developers can concentrate on software development, rather than environment setup. This leads to improved productivity [43].

A Managerial Level. Cloud supports managerial activities in terms of enhanced decision-making, increased standardization, and better customer relationship through extended features of cloud CRM software. A decision making process can benefit from centralized information and big data analytic tools to get insights from data within and beyond organizational boundary [29,42]. Cloud platforms can be used to uniform and streamline business processes among stakeholders across the value chains [29,42]. Productivity is expected to improve due to positive operational changes, leading to faster time-to-market in many business processes [34, 39]. In our view, productivity is classified as managerial rather than operational, as it requires longer time to manifest. Benefits from IT outsourcing are eligible for public cloud [25], including an opportunity to focus on core competencies [42] and to run business without technical employments [31].

A Strategic Level. Increased business agility is considered as a strategic advantage from cloud adoption [42]. Business agility is defined as the ability of organizations to flexibly and rapidly in response to changes in business environments [22]. Its supporting factors include minimum upfront investment [29], the use of IT as a competitive tool [34] and flexibility in production processes [36]

4.3 Finance

An Operational Level. In this category, benefits appear to be a consequence of efficient business operations [25,42], new approach to IT maintenances [37, 30,42], economies of scale [23, 25, 29, 37] and reduced energy consumption [26, 30]. The reduction on business operational cost is caused by the reduced cost per transaction [36], the ability to scale, and the standard systems and uniform processes [42].

A Managerial Level. As managerial activities involve planning, benefits of cloud manifest through a removal of long-term commitment and reduced IT cost variability. The usage-based pricing model provides an ability to pay for computing resources on a short-term basis, which in turn creates flexibility in financial planning [23, 31]. It also reduces cost variability of datacenters as it becomes unnecessary to prepare IT resources for unpredictable demand in advance [23, 38]. Lower real estate cost can be expected due to reduction in physical space requirement [25].

A Strategic Level. Financial benefits at the strategic level appear to be eligible for public clouds. This includes removal of upfront IT investment [23,30,42], an ability to transform capital expense to operational expense [25,28,32,34,38,41] and increased flexibility in investment distribution [33,34]. Such benefits play an essential role especially in a start-up context [33,34,42].

4.4 Innovation and Growth

An Operational Level. The innovation perspective contributes to sustainable success by supporting relevant factors for value creation, competitive advantage and continuous improvement. At the operational level cloud supports this objective by providing a mechanism for expanding product channels [25], technology integration and collaborations [25,33,41], enhanced analytical capability [34,42] and removal of non-value oriented work [25,31,38]. With cloud, new and trial services could be released to users easily, providing a cheap and convenient way for providers to gauge users' interests. Cloud platforms make it easier to combine enabling technologies to serve market requirements and to leverage development efforts of community. Outsourcing of manual work and higher automation of IT systems increase work satisfaction and accelerate creativity.

A Managerial Level. We consider the enhancements of resources to support innovation and value creation processes as a benefit of cloud at this level. This includes an extended pool of potential and motivated human resources [38], extended collaborative network [31,36] and extended IT resources [33,36, 39].

A Strategic Level. Benefits in this category appear as a consequence of innovation and value creation processes. Cloud adoption contributes to improved organizational image [39], enhanced variability and capability of new products [28,30,34,36,42,43] and increased competitive advantage [39,36,43]. Green organization and technological advancement improves the image of an organization in the eyes of the stakeholders. Enhancement of products and services based on cloud technology has been reported in many business areas including robotic [28], aircraft [30] and CRM [42]. Competitive advantage seems to be more relevant to start-ups as cloud removes the limitations on resources, expertise and investment.

4.5 Customers

In line with the work of Piotrowicz & Irani [10], the benefits toward customer satisfaction are considered as strategic. The literature mentions two main benefits. Better supports can be provided as cloud provides new channels to interact and engage with consumers [42]. Service quality is improved as unlimited IT resources can be acquired when demand increases [39].

5 Conclusions and Future Work

In this paper we built a benefit framework for cloud computing drawing upon the IS and cloud computing literature. The research contribution of this work is two folds. Firstly, the proposed conceptual framework captures the perspectives that should be considered when identifying the benefits of cloud computing in organizations. Secondly, the identified benefit items relevant to each perspective provide detailed guidelines for investigating and accessing cloud benefits in an organization in a more systematic manner. Our work has also practical implications for organizations. In an initial adoption phase, the framework serves as a source for evidences to make a more informed adoption decision. In a post-adoption phase, it can be applied to guide the benefit assessment. Future work should extend the framework by 1) using empirical evidences to verify the occurrences and the contexts of the benefits, 2) considering separately the benefits for different cloud services, and 3) defining benefit measures. A web based prototype for automating the use of the framework is accessible at https://www.inf.unibz.it/s4fs/index.php/projects/cloud-computing-benefit-framework.

References

1. Love, P.E.D., Irani, Z., Edwards, D.J.: Industry-centric benchmarking of information technology benefits, costs and risks for small-to-medium sized enterprises in construction. Automation in Construction 13, 507–524 (2004)
2. Farbey, B., Land, F., Targett, D.: The moving staircase: problems of appraisal and evaluation in a turbulent environment. Information Technology and People 12(3), 238–252 (1999)
3. Love, P.E.D., Irani, Z., Standing, C., Lin, C., Burn, J.M.: The enigma of evaluation: benefits, costs and risks of IT in Australian small-medium-sized enterprises. Information & Management 42(7), 947–964 (2005)

4. Irani, Z., Love, P.E.D.: The propagation of technology management taxonomies for evaluating information systems. Journal of Management Information Systems 17(3), 161–177 (2001)
5. Anthony, R.N.: Planning and Control Systems: A Framework for Analysis. Harvard University Press (1965)
6. Shang, S., Seddon, P.B.: Assessing and managing the benefits of enterprise systems: the business manager's perspective. Information Systems Journal 12, 271–299 (2002)
7. Schultheis, R., Sumner, M.: Management Information Systems: the Manager's View. Irwin, Boston (1989)
8. Farbey, B., Targett, D., Land, F.: Evaluating business information systems: reflections on an empirical study. Information Systems Journal 5, 235–252 (1995)
9. Kaplan, R.S., Norton, D.P.: The balanced scorecard–measures that drive performance. Harvard Business Review, 71–79 (January-February 1992)
10. Piotrowicz, W., Irani, Z.: Analysing B2B electronic procurement benefits: information systems perspective. Journal of Enterprise Information Management 23(4), 559–579 (2010)
11. Mingay, S., Furlonger, J., Magee, F., Andren, E.: The Five Pillars of Organizational Effectiveness. Gartner: R-06-6660 (1998)
12. Voas, J., Zhang, J.: Cloud Computing: New Wine or Just a New Bottle? IT Professional 11(2), 15–17 (2009)
13. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Department of Commerce. National Institute of Standards and Technology, U.S (2011)
14. Smith, D.M.: Hype Cycle for Cloud Computing. Gartner, Inc. (2012)
15. Truong, D.: How cloud computing enhances competitive advantages: a research model for small businesses. The Business Review 15(1), 59–65 (2010)
16. Dimensional research: Drivers of cloud adoption: A survey of CIOs and business executives. Sponsored by Host Analytics Inc . (2012)
17. Maluf, D.A., Shetye, S.D., Chilukuri, S., Sturken, I.: Lost in Cloud. In: IEEE Aerospace Conference, pp. 1–6 (2012)
18. Phaphoom, N., Oza, N., Wang, X., Abrahamsson, P.: Does cloud computing deliver the promised benefits for IT industry? In: Proceedings of the WICSA/ECSA 2012 Companion Volume, pp. 45–52. ACM (2012)
19. Deed, C., Cragg, P.: Business Impacts of Cloud Computing. Cloud Computing Service and Deployment Models: Layers and Management, pp. 274–288. IGI Global (2013)
20. Duncan, N.B.: Capturing flexibility for information technology infrastructure: a study of resource characteristics and their measure. Journal of Management Information Systems 12, 37–57 (1995)
21. Phaphoom, N., Wang, X., Abrahamsson, P.: Foundations and technological landscape of cloud computing. International Scholarly Network in Software Engineering (2012)
22. Sambamurthy, V., Bharadwaj, A., Grover, V.: Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. MIS Quarterly 27(2), 237–263 (2003)
23. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010)
24. Barcomb, K.E., Humphries, J.W., Mills, R.F.: A case for DoD application of public cloud computing services. In: Military Communications Conference, pp. 1888–1893 (2011)
25. Khajeh-Hosseini, A., Sommerville, I., Bogaerts, J., Teregowda, P.: Decision Support Tools for Cloud Migration in the Enterprise. In: IEEE International Conference on Cloud Computing, pp. 541–548 (2011)

26. Li, C., Deng, Z.: Value of Cloud Computing by the View of Information Resources. In: International Conference on Network Computing and Information Security, vol. 1, pp. 108–112. IEEE Computer Society, USA (2011)
27. Liang, D.-H., Liang, D.-S., Chang, C.-P.: Cloud Computing and Green Management. In: International Conference on Intelligent System Design and Engineering Application, pp. 639–642 (2012)
28. Abidi, F.: Cloud computing and its effects on healthcare, robotics, and piracy. In: World Congress on Sustainable Technologies, pp. 135–140 (2011)
29. Heier, H., Borgman, H.P., Bahli, B.: Cloudrise: Opportunities and Challenges for IT Governance at the Dawn of Cloud Computing. In: Hawaii International Conference on System Science, pp. 4982–4991 (2012)
30. Jasti, A., Mohapatra, S., Potluri, B., Pendse, R.: Cloud computing in Aircraft Data Network. In: Integrated Communications, Navigation and Surveilance Conference, pp. E7:1-8 (2011)
31. Juliandri, A., Musida, M.: Supriyadi: Positioning cloud computing in machine to machine business models. In: International Conference on Cloud Computing and Social Networking, pp. 1–4 (2012)
32. Kaisler, S., Money, W.H., Cohen, S.J.: A Decision Framework for Cloud Computing. In: Hawaii International Conference on System Science, pp. 1553–1562 (2012)
33. Xin, L., Song, C.: Cloud-based innovation of Internet long tail. In: International Conference on Product Innovation Management, pp. 603–607 (2011)
34. Marston, S., Zhi, L., Bandyopadhyay, S., Ghalsasi, A.: Cloud Computing - The Business Perspective. In: Hawaii International Conference on System Sciences, pp. 1–11 (2011)
35. McDonald, D., MacDonald, A., Breslin, C.: Review of the environmental and organisational implications of cloud computing in higher and further education. University of Strathclyde (2010)
36. Mladenow, A., Fuchs, E., Dohmen, P., Strauss, C.: Value Creation Using Clouds: Analysis of Value Drivers for Start-Ups and Small and Medium Sized Enterprises in the Textile Industry. In: International Conference on Advanced Information Networking and Applications Workshops, pp. 1215–1220 (2012)
37. Rafique, K., Tareen, A.W., Saeed, M., Jingzhu, W., Qureshi, S.S.: Cloud computing economics opportunities and challenges. In: IEEE International Conference on Broadband Network and Multimedia Technology, pp. 401–406 (2011)
38. Rayport, J.F., Heyward, A.: Envisioning the Cloud: The Next Computing Paradigm. Marketspace LLC (2009)
39. Wu, Z., Gan, A.: Qualitative and Quantitative Analysis the Value of Cloud Computing. In: International Conference on Information Management, Innovation Management and Industrial Engineering, vol. 2, pp. 518–521 (2011)
40. ENISA: Cloud Computing: Benefits, Risks and Recommendations for Information Security. European Network and Information Security Agency (2009)
41. Goodburn, M.A.: The cloud transforms business. Financial Executive 22, 1–6 (2010)
42. Harris, J.G., Alter, A.E.: Cloudrise: rewards and risks at the dawn of cloud computing. Accenture Institute for High Performance, Chicago (2010)
43. Zhang, K.-J., Ma, B., Dong, P.-J., Tang, B.-Y., Cai, H.: Research on Producer Service Innovation in Home-Textile Industrial Cluster Based on Cloud Computing Platform. In: International Conference on Service Operations and Logistics and Informatics, pp. 155–160. IEEE Press (2010)

The Importance of the Business Idea for New Venture Creation in the Software Industry

Natalie Kaltenecker , Christian Hoerndlein, and Thomas Hess

Ludwig-Maximilians-Universität München, Institute for Information Systems and New Media, Ludwigstraße 28, 80539 Munich, Germany
{kaltenecker,hoerndlein,thess}@bwl.lmu.de

Abstract. Entrepreneurial activity, especially the creation of new companies in the growing information and communication technology sector, is of high importance to foster an economy's growth. However, despite calls for research to investigate why young people decide to start their own business, there is a lack of research to identify the role of the Business Idea in this context. In this paper, we conceptualize the construct "Business Idea" and test its influence on the intention to start a company in the software industry by drawing on the Theory of Planned Behavior. Empirical evidence from a survey among information technology students (N=402) shows that the Business Idea is a major driver for the intention to found a company. The study thus contributes to a better understanding of the factors driving the intention to start a new company in general and the Business Idea's importance in particular.

Keywords: Business Idea, Entrepreneurship, Software Industry.

1 Introduction

Entrepreneurial activity is an important element for generating economic growth [1]. According to Zoltan and Audretsch [2], entrepreneurship contributes to the creation of new products and services. Young technology-focused companies create the jobs of tomorrow. Therefore, more people should dare to be entrepreneurs. But what are the factors that determine a person's decision to create a new venture?

Especially the information and communication technology (ICT) industry is seen as a key to economic growth [3]. ICT forms the technological foundation of the information and knowledge society. It permeates all aspects of our lives and has become the main driver of innovation. Furthermore, the ICT industry is seen as a key enabling factor for other industries and is recognized as bringing productivity improvements and sustainable competitive advantage to a nation's companies [4]. Within the field of ICT, software has evolved as one of the most important aspects because of its ubiquity and its flexibility [5].

Given the importance of the topics entrepreneurship and ICT, it is surprising that research usually focuses either on one topic or the other. On the one hand, there seems to be a lack concerning entrepreneurship research which focuses on the ICT industry.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 153–165, 2013.

Studies foremost investigate whether total entrepreneurial activity influences countries' growth of the gross domestic product [1, 6] and do not explore the factors that contribute to the foundation of new ICT ventures. On the other hand, research specific to the ICT industry usually neglects entrepreneurial activity. Rather, researchers assume an ICT company to exist already, and focus on the challenges of an already-operating company, such as technology design [7] or pricing [8].

As part of a related research project we conducted qualitative interviews with about 30 founders of software companies. Instead of seed capital, technical knowledge, or the services offered in new-venture incubators, one major theme that emerged as central for starting a new company was the Business Idea. Creativity and the entrepreneurial idea seem to play a central role when it comes to the intention to found a company in the software sector. However, the Business Idea plays only a minor role in extant research, and there is a lack of studies that examine its impact. This research gaps serves as a starting point for our research in this paper.

In existing studies, the Business Idea is either not dealt with at all [9] or is presumed to be a given factor [10]. Besides, despite calls from politics for more entrepreneurial activity in the software industry [e.g. 11], existing research focuses on potential founders with a business background [12]. Instead, we want to focus on technically-oriented information technology students who are a likely source of venture creation. From these aspects, we derive our main research question:

What factors can be identified that influence the intention to found a company among information technology students, and what is the specific role of the Business Idea?

Through the introduction of the concept of the Business Idea, this paper fosters an interdisciplinary perspective between ICT and entrepreneurship. In our conception, the Business Idea bridges the realms of the rather technically oriented ICT research and the more socially focused research in entrepreneurship. In our study, we apply Shane and Venkataraman's [13] considerations that an objective entrepreneurial opportunity exists, which has to be discovered subjectively. We perceive of a technically-oriented opportunity in the software industry which has to be discovered by an entrepreneur to be turned into a Business Idea. Thus, our study has the potential to open up new streams of scientific inquiry by connecting so far only loosely connected research fields.

The remainder of this study is structured as follows: First, we will develop a theoretical framework and derive our propositions. Subsequently, we will test this framework through an empirical study among information technology students. We will conclude this paper by pointing out the practical implications and considering the potential limitations of this study.

2 Theoretical Development

Entrepreneurship has been under investigation in various strands of literature. In the following section, we summarize the relevant literature and develop a research model based on this analysis. Following these steps, we derive our six hypotheses.

2.1 Definition

Given the abundance of scientific fields that deal with entrepreneurship research, it is difficult to derive an all-embracing definition of "entrepreneurship". Within our study, entrepreneurship refers on the one hand to the creation of a new company in the software industry and not the separation from or consolidation of already existing companies [14]. Furthermore, our study's focus is on the so-called "opportunity entrepreneurship" [15, p. 11], which means that the target group (information technology students) develops entrepreneurial intentions to realize chances and to fulfill their own ideas concerning a software product or a software service. In the literature this motivation to establish a company is often termed "pull" motivation [15]. In contrast to the "push" motivation, in which entrepreneurship occurs due to misery respectively imminent unemployment, the "pull" motivation is growth-oriented and provides more new jobs [16].

From this point of view, our conception of an entrepreneur is based on the behavioral-oriented definition [15]. It states that "entrepreneurship is concerned with the discovery and exploitation of profitable opportunities" [15, p. 10]. Our focus in this study will be on the software industry, more specifically on the development of a software product or a software-based service; not included are companies offering exclusively training courses and consulting services for information systems.

2.2 Related Work

In this section, we introduce several theoretical concepts, which give a short overview of the state of the art in entrepreneurship research.

Intention-based models have evolved to be the most widely-used type of models to explain entrepreneurial behavior. In this type of models, the intention moderates the influence of specific factors on actual behavior. One of these intention-based models is Ajzen's [17] Theory of Planned Behavior. Despite other alternative intention-based models in the field of entrepreneurship research, such as the "Shapero-Krueger Model of Entrepreneurial Intent" [18], Ajzen's theory [17] has become widely accepted in entrepreneurship research.

The Theory of Planned Behavior has been empirically tested in the general field of entrepreneurship, but until now there has been a lack of research on the role of the Business Idea in particular. Embedding this substantially new explanation factor and its impact on the intention to found a company will be the main contribution of this paper.

Possibly the most important and most prominent article concerning current entrepreneurship research has been written by Shane and Venkataraman [13, 19, 20]. With their entrepreneurship framework they explain a set of empirical phenomena and predict a set of outcomes not explained or predicted by conceptual frameworks already used in other fields. The authors subdivide the entrepreneurial opportunity in three different aspects: the existence, the discovery and the exploitation of

entrepreneurial opportunities. While the discovery of an entrepreneurial opportunity is a subjectively controlled process, the opportunity itself is an objective phenomenon which existence is however not known to everybody.

In existing studies, the Business Idea has not been operationalized. In particular, these studies do not ask potential founders if they have a Business Idea which possibly explains their intention to create their own business. Rather, these studies analyze if (potential) founders think that they can realize their own ideas after having founded their own company. The focus is therefore not on having a Business Idea that leads to the intention to establish an own firm but it is more connected to the freedom of working on one's own ideas in an own company [21, 14].

According to Klofsten [22], however, "[o]ne of the requirements for starting a firm is an idea that can be developed into a business opportunity" [22, p. 195]. Having an idea is therefore a prerequisite concerning the establishment of a firm. The difference between his and our study is the operationalization: In his work, he interviewed five founders, who in hindsight judged the factors that influenced their decision to create their own company. One of the main factors that emerged was the Business Idea. We, in contrast, analyze the Business Idea's influence on the intention to found a company from an ex-ante perspective.

In summary, there is a difference between existing studies and our understanding of a Business Idea as well as the way we plan to operationalize it. Thus, our paper attempts to close the illustrated research gaps.

2.3 Research Model and Hypotheses Development

We will use Ajzen's [17] Theory of Planned Behavior as theoretical framework for our study and expand it with the newly developed construct "Business Idea". Within the scope of this model we act on the assumption that Intention represents the best predictor for actual behavior [23].

Figure 1 shows our developed research model. We point out that the Theory of Planned Behavior is fully[1] established within the model. Attitude, Subjective Norm and Perceived Behavioral Control have a direct positive effect on the Intention to found a company. Based on our literature analysis, we propose that the Business Idea's positive influence on Intention is mediated through Attitude and Perceived Behavioral Control. Besides, bearing the qualitative interviews of our related research project in mind, we suppose that the Business Idea also has a direct positive effect on Intention[2].

---

[1] Without the bidirectional influence among Attitude, Subjective Norm and Perceived Behavioral Control.

[2] Solid lines indicate the Theory of Planned Behavior (H_1, H_2, H_3); dotted lines indicate the new propositions integrated into the Theory of Planned Behavior (H_4, H_5, H_6).

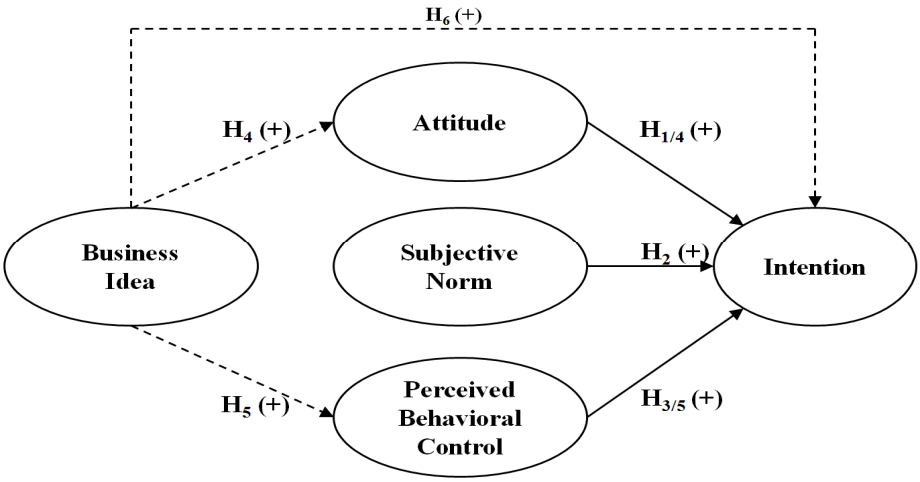

Fig. 1. Research Model

We will now further elaborate on our research hypotheses.

Intention is seen as a direct determinant of human behavior. Ajzen [17] describes Intentions as follows: "Intentions are assumed to capture the motivational factors that influence a behavior" [17, p. 181]. The Theory of Planned Behavior postulates three conceptually independent determinants of a person's Intention: Attitude, Subjective Norm, and Perceived Behavioral Control.

First, Attitude towards a certain behavior refers to the degree to which a person has a favorable or unfavorable evaluation or appraisal of the respective behavior. The second predictor is a social factor termed Subjective Norm. It refers to the perceived social pressure to perform or not to perform the behavior. The third antecedent of one's Intention is the degree of Perceived Behavioral Control, which refers to the perceived ease or difficulty of performing the behavior. It is assumed to reflect past experience as well as anticipated impediments and obstacles. As a general rule, the more favorable the Attitude, the stronger the Subjective Norm, and the greater the Perceived Behavioral Control with respect to a certain behavior, the stronger an individual's Intention to perform this behavior [17].

Building upon existing research, we formulate the following three hypotheses to apply the Theory of Planned Behavior in our specific context of entrepreneurial Intentions:

H_1: There is a positive relationship between Attitude and the Intention to start a new business.
H_2: There is a positive relationship between Subjective Norm and the Intention to start a new business.
H_3: There is a positive relationship between Perceived Behavioral Control and the Intention to start a new business.

Following Ajzen's Theory of Planned Behavior, Attitude, Subjective Norm and Perceived Behavioral Control are the only determinants of one's Intention [17].

Therefore, any additional variable's impact, such as the Business Ideas' influence, on Intention could only be mediated through these three constructs. Based on the analysis of the literature which deals with the concept and definition of an idea in general, we assume a potential mediator effect through the construct Attitude [24, 21] as well as through the construct Perceived Behavioral Control [25, 26, 24, 22, 14]. This view is supported by the analysis of our qualitative interviews, which reveals that most people have trust in their own ideas and also value the risk of their own ideas failing lower. Therefore the Business Idea has a positive effect on someone's Attitude. In terms of the Perceived Behavioral Control, the Business Idea can be regarded as a key factor in feeling able to actually create an enterprise. However, we could not derive an influence of the Business Idea on the Subjective Norm from our literature analysis.

Therefore, we formulate the following hypotheses:

H_4: The positive relationship between the Business Idea and the Intention to start a new business is mediated through the Attitude.

H_5: The positive relationship between the Business Idea and the Intention to start up a new business is mediated through the construct of Perceived Behavioral Control.

The sixth hypothesis results from the analysis of the qualitative interviews with founders of software companies. The Business Idea played a crucial role regarding the founding of their companies. Their Business Idea served as a vision and driver concerning the Intention to start a new business, independent from any indirect influence on Attitude or Perceived Behavioral Control. In line with Klofsten [22], the presence of a Business Idea had been a necessary condition to develop the Intention to found a company.

Therefore, we derive the following hypothesis:

H_6: There is a positive relationship between the Business Idea and the Intention to start a new business.

3 Scale Development Procedure and Main Survey

Recognizing the lack of previous scale development efforts concerning entrepreneurial ideas, we conceptualize a construct called "Business Idea" and test its influence on the Intention to start a company in the software industry by drawing on the Theory of Planned Behavior. We followed a three-step process to develop, refine, and validate the measurement model and test our hypotheses. These three steps were (1) the conceptualization and development of indicators, (2) the refinement of indicators, and (3) the main survey to validate the measurement instrument and our research model.

3.1 Step 1: Conceptualization and Development of Indicators

With regard to the content specification of the constructs, i.e. the concrete choice of indicators, this step consists of several components [27, 28, 29].

Regarding the generation of the reflective items, we have to differentiate between the already available and validated constructs in the literature, namely Attitude, Subjective Norm, Perceived Behavioral Control and Intention, and on the other hand our newly developed construct "Business Idea". Concerning the former ones, we have adopted reflective constructs from the literature [30].

In terms of the newly developed construct "Business Idea", we created seven reflective items based on prior literature and founders' statements in the qualitative interviews. Since we could not revert to an already verified construct in literature, the unidimensionality and reliability of these items were tested with a pre-test (see step 2). Table 1 shows the seven items for the construct "Business Ideas".

Table 1. Items for the Construct "Business Idea"

Item 1	I have an idea to start a company in the software industry.
Item 2	I have discovered a market niche in the software industry.
Item 3	I have an innovative idea for the software industry.
Item 4	I have an idea for a software product or a software service.
Item 5	I have an entrepreneurial idea for the software industry.
Item 6	I have a business idea for the software industry.
Item 7	I have had an inspiration for a software product or a software service.

3.2 Step 2: Refinement of Indicators

A first pre-test (N=21) was undertaken in December 2011 to test the developed questionnaire in preparation of the main survey and to ensure that all the items were understood as intended. The pre-test generated positive feedback concerning length and comprehensibility. Except for small adjustments (e.g. reducing the Likert scale from seven to five points), we maintained the basic structure of the survey. Additionally, principal component analysis showed that all seven reflective items to measure the construct "Business Idea" load on one single factor.

3.3 Step 3: Main Survey

We distributed the 2-page questionnaire to information technology students from three public universities[3] during the winter term 2011/2012. We asked students from nine different courses on a bachelor and master level to participate in the survey. Data was collected during January and February 2012. The decision in favor of a paper-based and not an online survey was made based on a higher expected response rate [31, 32]. A total of 598 questionnaires were distributed and 402 completed responses were received, yielding a response rate of 67.2%. This rate reflects the respondents' high interest towards the research subject. The sample (N=402) consists of 295 (73.4%) male and 107 (26.6%) female students within the disciplines of Information Systems (44.3%), Media Informatics (27.9%) and Computer Science (20.4%). A

[3] For the purpose of this blind review, we have not indicated the names of the three universities.

further 7.4% of the respondents had chosen Computer Science as a minor. 24.4% of the respondents pursued a master's and 75.6% a bachelor's degree. Along with the five constructs that we have specified above, the questionnaire included control variables on the students' background and demographics [33].

Concerning the following statistical analysis, we first assessed the reflective measurement model and subsequently the path model. For the analysis of the path coefficients, we used Partial Least Square (PLS) path modeling.

3.4 Analysis of the Measurement Model

The measurement model was assessed in terms of the following local quality criteria: composite reliability, average variance extracted (AVE) and discriminant validity [34]. Concerning the scores for the composite reliability, the values lie between 0.90 and 0.96 and thus exceed the recommended threshold value of 0.7 [35]. Looking at the values for Cronbach's Alpha, they are also comfortable above the suggested threshold value of 0.7 [35].

Table 2 shows the inter-construct correlations, with the AVE values on the matrix's diagonal. All square roots of the AVE values exceed inter-construct correlations, thus providing strong evidence of discriminant validity [34].

Table 2. Inter-construct Correlations and AVE

	Attitude	Business Idea	Intention	Subjective Norm	Perceived Behavioral Control
Attitude	0.75	-	-	-	-
Business Idea	0.33	0.79	-	-	-
Intention	0.55	0.61	0.90	-	-
Subjective Norm	0.46	0.21	0.32	0.83	-
Perceived Behavioral Control	0.22	0.31	0.37	0.20	0.77

Because of the strong correlation between the constructs Business Idea and Intention, we also checked whether the respective items are actually tapping into two different constructs. Our analysis shows that the maximum of inter-construct item correlations is smaller than the minimum of the intra-construct item correlations of the two constructs. We therefore conclude that the items that we developed to measure Business Idea are sufficiently distinct from the items to measure Intention.

During the course of our study, we received feedback whether having a Business Idea should not better be measured using a binary variable. We analyzed the distribution of the Business Idea's values in the survey; it follows approximately an equal distribution. Therefore, we can conclude that people have certain degrees of a Business Idea instead of having a Business Idea or not.

3.5 Analysis of the Path Model

Consistent with established recommendations on the analysis of structural equation models (SEMs), values for the explained variance (R^2), effect sizes (f^2) and the t-values of the bootstrapping procedure [36, 37] were calculated. The values were generated by or based on the output generated by the SmartPLS 2.0 [38] software[4]. According to Hair et al. [39], PLS-SEM analysis is highly recommended for explorative analysis, as is the case in our study.

The explained variance in the model is 52.7% for the Intention to start a new company in the software industry, which can be interpreted as the model showing medium to substantial explanatory power [40]. Furthermore, the Business Idea explains 11.2% of the Attitude's variance and 9.4% of the Perceived Behavioral Control's variance.

Bootstrap values are a way for measuring the statistical significance of an estimated coefficient path coefficient in PLS, which has no distributional assumptions [39]. Based on the calculated t-values we can determine the significance of a specific parameter [41]. Table 3 shows the path coefficients along with their respective significance level.

Table 3. Path Coefficients and Significance

Paths	Path coefficients
Attitude→Intention (H_1)	0.348***
Subjective Norm→Intention (H_2)	0.046
Perceived Behavioral Control→ Intention (H_3)	0.145***
Business Idea→Attitude (H_4)	0.334***
Business Idea→Perceived Behavioral Control (H_5)	0.307***
Business Idea→Intention (H_6)	0.440***

(*** $p<0.001$)

Except for the influence of the Subjective Norm on the Intention, all path coefficients were found to be significant at a 0.1%-level. A Sobel Test revealed that Attitude and Perceived Behavioral Control both partially mediate the Business Idea's influence on Intention [42]. The analysis therefore confirms the hypotheses H_1, H_3, H_4, H_5 and H_6, but we have to reject H_2.

The calculations concerning the effect sizes regarding Intention's variance show a minor influence of the Perceived Behavioral Control (f^2=0.032) and a medium effect size of the Attitude (f^2=0.184). With an f^2 value of 0.340, the Business Idea has the strongest influence on the Intention, which is characterized in literature as a medium to strong influence [40].

If we exclude the Business Idea from the model, which results in the Theory of Planned Behavior's base model, 36.6% of the Intention's variance can be explained. This indicates that in the case of entrepreneurship, the Theory of Planned Behavior might benefit from including the Business Idea as a construct, which contributes to a considerable increase in the explained variance (36.6% vs. 52.7%).

[4] PLS Algorithm: Case Wise Replacement, Path Weighting Scheme, Mean 0, Var 1; Bootstrapping Algorithm: Case Wise Replacement, Individual Changes (N=383).

4 Discussion, Limitations, and Further Research

Our study confirms the importance of the entrepreneurial Business Idea's impact on the intention to start a new company: By including this construct into the Theory of Planned Behavior, we are able to explain an additional 16.1 percentage points of the behavioral intention's variance compared to the Theory of Planned Behavior's baseline model. The results show that the construct of the Business Idea captures aspects that are not included in the original Theory of Planned Behavior's model and whose effects on Intention are not mediated by Attitude or Perceived Behavior Control. In the field of entrepreneurial research, the Theory of Planned Behavior model should therefore be extended.

The study's findings have important policy implications: The Business Idea has the biggest impact on IT students' intention to found a company. Therefore, efforts to increase the entrepreneurial activity in the software industry should focus on creative thinking and opportunity detection. This might seem counterintuitive at first: The software industry is probably associated more with rather "hard" factors such as technological knowledge or computing equipment and to a lesser extent with "soft" factors. Of course, these factors are also important and should not be neglected, but it is the factor of having a Business Idea that has the biggest impact on starting a new venture.

These results resonate with a shift of how IT is perceived in academic research. Instead of regarding IT and its use and adoption as a purely technological artifact, research has started paying more attention to the notion that "[t]echnologies are simultaneously social and physical artifacts" [43, p. 149]. Our results show that also entrepreneurship should be seen as being embedded within a social system, in which creativity and other social factors have a crucial impact on the intention to found a company. Therefore, to foster entrepreneurial activity, especially in the ICT sector, it needs more than venture capital and technical know-how, but rather a creative environment in which entrepreneurial ideas can prosper.

We suggest that further research in ICT should focus on how new technologies can be exploited and turned into Business Ideas, which are the basis for new ICT companies. This might include revising information technology curricula to encourage students to exploit the technical knowledge that they are taught. By framing current topics such as Cloud Computing not only as a new technology but also as an opportunity for new venture creation, universities might actively contribute to a country's entrepreneurial activity. Having interdisciplinary teams work together on IT projects during their courses could also stimulate creativity and unconventional thinking. Besides, IT students could be required to attend lectures from non-IT fields. By applying knowledge and methods from other disciplines to their own field of study, IT students might develop Business Ideas that they would not have come up without thinking "outside the box".

There are some potential limitations that apply to our study: First of all, we restricted our survey to information technology students. Although this group is an important source of new-venture creation in the software industry, we excluded other groups such as students from other disciplines. Besides, our study's conclusions are

based on the questionnaires that we distributed in three public universities. Further studies should include a wider range of academic institutions and also include institutions from different countries.

Second, by conducting a cross-sectional study, we could only collect data regarding the intention to found a new company. However, one could argue that in processes with a lot of decision points and impact factors, such as founding a company, the intention might only weakly predict actual behavior.

Third, our study focuses on the "opportunity recognition" and "opportunity discovery". The "opportunity creation" is outside the scope of our study [44].

In summary, our study can serve as a starting point for deepening the understanding why people decide to found a company. Our findings stress the importance of the Business Idea, a construct that had been neglected in extant research. We encourage other researchers to replicate our study in other settings to verify the relationships that we found to be significant.

References

1. van Stel, A., Carree, M., Thurik, A.R.: The effect of entrepreneurial activity on national economic growth. Max Planck Institute for Research into Economic Systems, Jena, pp. 1–22 (2005)
2. Zoltan, J., Audretsch, D.: Handbook of Entrepreneurship Research: An Interdisciplinary Survey and Introduction. Springer, New York (2010)
3. van Oort, F.G., Stam, E.: Agglomeration Economies and Entrepreneurship in the ICT Industry. Research in Management 16, 1–24 (2006)
4. Underwood, J., Khosrowshahi, F.: ICT Expenditure and Trends in the UK Construction Industry and Facing the Challenges of the Global Economic Crisis. Journal of Information Technology in Construction 17, 25–42 (2011)
5. Tessler, S., Barr, A., Hanna, N.: National Software Industry Development: Considerations for Gevernment Planners. The Electronic Journal of Information Systems in Developing Countries 13(10), 1–17 (2003)
6. Audretsch, D., Keilbach, M.: Entrepreneurship Capital and Economic Performance. Regional Studies 38, 949–959 (2004)
7. Erl, T.: Service-Oriented Design - Part IV: Business Process Design. In: Erl, T. (ed.) Service-Oriented Architecture: Concepts, Technology, and Design, pp. 565–611. Pearson Education, Inc., New York (2005)
8. Ayyagari, M., Demirgüc-Kunt, A., Maksimovic, V.: Firm Innovation in Emerging Markets: The Role of Finance, Governance, and Competition. Journal of Financial and Quantitative Analysis 46(6), 1545–1580 (2011)
9. Franke, N., Lüthje, C.: Studentische Unternehmensgründungen – dank oder trotz Förderung? Schmalenbachs Zeitschrift für Betriebswirtschaftliche Forschung 54(3), 96–112 (2002)
10. Halberstadt, J., Ossietzky, C., Welpe, I.: Motive, Eigenschaften und Emotionen von Unternehmensgründern. In: Kraus, S., Fink, M. (eds.) Entrepreneurship - Theorie und Fallstudien zu Gründungs-, Wachstums- und KMU-Management, pp. 52–67. WUV Facultas, Wien (2008)
11. Federal Ministry of Economics and Industry: Rückblick: Gründerwoche Deutschland 2011. Federal Ministry of Economics and Industry, Berlin (2012)

12. Weber, S., Starke, S.: Lernpotenzial und Effekte eines Business Planning-Kurses. Unterrichtswissenschaft 38, 292–317 (2010)
13. Shane, S., Venkataraman, S.: The Promise of Entrepreneurship as a Field of Research. The Academy of Management Review 25(1), 217–226 (2000)
14. Tegtmeier, S.: Die Existenzgründungsabsicht - Eine theoretische und empirische Analyse auf Basis der Theory of Planned Behavior. Tectum Verlag Marburg, Lüneburg (2008)
15. Stephan, U.: Culture of Entrepreneurship (C-ENT)/Kultur der Selbständigkeit: Konzeptualisierung und erste Validierung eines Fragebogens zur Erfassung einer unternehmertumsförderlichen Kultur. Psychologie, Philipps-Universität Marburg, Marburg (2008)
16. Llisterri, J.J., Kantis, H., Angelelli, P., Tejerina, L.: Is Youth Entrepreneurship a Necessity or an Opportunity? A First Exploration of Household and New Enterprise Surveys in Latin America (2006)
17. Ajzen, I.: The Theory of Planned Behavior. Organizational Behavior and Human Decision Processes 50, 179–211 (1991)
18. Krueger, N.: Entrepreneurial Intentions are Dead: Long Live Entrepreneurial Intentions. In: Carsrud, A., Brännback, M. (eds.) Understanding the Entrepreneurial Mind, pp. 51–75. Springer, New York (2009)
19. Shane, S.: Reflections on the 2010 AMR Decade Award: Delivering on the Promise of Entrepreneurship as a Field of Research. Academy of Management Review 37(1), 10–20 (2012)
20. Venkataraman, S., Sarasvathy, S., Dew, N., Forster, W.: Reflections on the 2010 AMR Decade Award: Whither the Promise? Moving forward with Entrepreneurship as a science of the Artificial. Academy of Management Review 37(1), 21–33 (2012)
21. Jacob, K.: Unternehmer aus Hochschulen? Eine Studie zu Existenzgründungsabsichten von Studierenden. Verlag Dr. Kovac, Hamburg (2007)
22. Klofsten, M.: New Venture Ideas: An Analysis of their Origin and Early Development. Technology Analysis & Strategic Management 17(1), 105–119 (2005)
23. Bamberg, S.: Helfen Implementationsintentionen, die Lücke zwischen Absicht und Verhalten zu überwinden? Ergebnisse zweier interventionsorientierter Feldexperimente. Zeitschrift für Sozialpsychologie 33(3), 143–155 (2002)
24. Funke, J.: Psychologie der Kreativität. In: Holm-Hadulla, R.M. (ed.) Kreativität, pp. 283–300. Springer, Heidelberg (2000)
25. Bird, B.: Implementing Entrepreneurial Ideas: The Case for Intention. Academy of Management Review 13(3), 442–453 (1988)
26. Eagly, A.H., Chaiken, S.: The psychology of attitudes. Harcourt Brace Jovanovich College Publishers, Fort Worth (1993)
27. Churchill, G.: A Paradigm for Developing Better Measures of Marketing Constructs. Journal of Marketing Research 16(1), 64–73 (1979)
28. DeVellis, R.F.: Scale Development - Theory and Applications. Sage, Thousand Oaks (2003)
29. Hinkin, T.R.: A Brief Tutorial on the Development of Measures for Use in Survey Questionnaires. Organizational Research Methods 1(1), 104–121 (1998)
30. Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D.: User Acceptance of Information Technology: Toward a Unified View. Management Information Systems Quarterly 27(3), 425–478 (2003)
31. Friedrichs, J.: Methoden empirischer Sozialforschung. Opladen (1980)
32. Schnell, R., Hill, P.B., Esser, E.: Methoden der empirischen Sozialforschung. Oldenburg, München (2005)

33. Fishbein, M., Ajzen, I.: Predicting and changing behavior: The reasoned action approach. Psychology Press, New York (2010)
34. Fornell, C., Larcker, D.F.: Evaluating Structural Equation Models with Unobservable Variables and Measurement Error. Journal of Marketing Research 18(1), 39–50 (1981)
35. Hair, J.F., Black, W.C., Babin, B.J., Anderson, R.E., Tatham, R.L.: Multivariate Data Analysis. Pearson Prentice Hall, New Jersey (2006)
36. Bliemel, F., Eggert, A., Fassott, G., Henseler, J.: Handbuch PLS Pfadmodellierung - Methode Anwendung Praxisbeispiele. Schäffer-Poeschel, Stuttgart (2005)
37. Gefen, D., Rigdon, E., Straub, D.: An Update and Extension to SEM Guidelines for Administrative and Social Science Research. Journal of Management Information Systems 35(2), iii–xiv (2011)
38. Ringle, C.M., Wende, S., Sinkovics, R.R.: SmartPLS 2.0 (2005), http://smartpls.de
39. Hair, J.F., Ringle, C.M., Sarstedt, M.: PLS-SEM. Indeed a Silver Bullet. Journal of Marketing Theory and Practice 19(2), 139–151 (2011)
40. Vinzi, V.E., Trinchera, L., Amato, S.: PLS Path Modeling: From Foundations to Recent Developments and Open Issues for Model Assessment and Improvement. In: Vinzi, V.E. (ed.) Handbook of Partial Least Squares, pp. 47–82. Springer, Heidelberg (2010)
41. Tenenhaus, E., Vinzi, V.E., Chatelinc, Y.M., Laurob, C.: PLS path modeling. Computational Statistics & Data Analysis 48, 159–205 (2004)
42. Urban, D., Mayerl, J.: Mediator-Effekte in der Regressionsanalyse direkte, indirekte und totale Effekte. University of Stuttgart (2007)
43. Orlikowski, W.J., Barley, S.R.: Technology and Institutions: What can Research on Information Technology and Research on Organizations Learn from each other? MIS Quarterly 25(2), 145–165 (2001)
44. Sarasvathy, S.D., Dew, N., Velamuri, R., Venkataraman, S.: Three Views of Entrepreneurial Opportunity. In: Acs, Z.J., Audretsch, D.B. (eds.) Handbook of Entrepreneurship Research, 2nd edn. International Handbook Series on Entrepreneurship, vol. 5, pp. 77–96. Springer, Heidelberg (2010)

Exploring How Feature Usage Relates to Customer Perceived Value: A Case Study in a Startup Company

Sarunas Marciuska, Cigdem Gencel, and Pekka Abrahamsson

Free University of Bolzano-Bozen,
Marciuska@inf.unibz.it, {Cigdem.Gencel,Pekka.Abrahamsson}@unibz.it

Abstract. Most of the business value of a software product comes from only a small proportion of its features. Product managers face the challenge of identifying the high value features in an application and weeding out the ones of low value from the next releases. What creates this challenge is the fact that customer perceived value is an attribute, dimensions of which are not well-known yet. Currently, software companies try to assess the value of features through interviewing a number of key stakeholders. However, the literature suggests that, this kind of evaluation could be misleading due to stakeholders having different understanding of what value refers to. In this paper, through an exploratory case study, we investigate how usage of features relate to their perceived value and shed light into the factors affecting this relationship. The results show that feature usage metric has a significant potential to estimate value of features.

Keywords: Value based software engineering, Customer perceived value, Feature usage.

1 Introduction

Value based software engineering [1,2] has gained considerable attention in the last years. Software companies have been trying to define mechanisms for integrating value-based decision making in their product development and evolution life cycle in order to sustain growth and maintain competitive advantage [1,3].

Value-based decision making helps in developing high quality products on time and within budget by taking right strategic decisions considering the benefit, value and risk factors throughout the life cycle [4]. However, there are a number of challenges the practitioners are facing [1,2,5,6] –How to measure and/or estimate software value that has many dimensions? and –Which of these dimensions (i.e. customer perceived value, product value, competitive advantage value, innovation value) are important to consider in decision making? [7,8,9,10].

In this paper, our focus is on customer perceived value. It is stated that eighty percent of business value comes from only twenty percent of software components [11]. Here, the problem is two-fold: 1) estimating the features that would increase customer perceived value during the requirements analysis phase

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 166–177, 2013.

and prioritising them for the next release, and 2) monitoring the perceived value of features of a product that is already in use, and identifying and weeding out the ones that are of low value during the next release of the product.

The first challenge has been studied considerably for market-driven development under the requirements prioritisation area. Several methods have been used in industry such as Analytical Hierarchy Process [12], 100-point method [13], Planning game [14] etc. In this paper, however, we deal with the second challenge of identifying features of a product in use, which are not so valuable from the customers' point of view.

Recent studies suggest that high customer perceived value is the key for creating long-term industrial relationships, which also can impact the product value [15]. There are a number of techniques developed to measure customer perceived value [16,17,18]. Most of them are based on making interviews with key customers to get their subjective opinion and generalise the results for a population. However, this approach has some major challenges. First, it is difficult for customers to estimate customer perceived value because of its multidimensional nature. Second, it may be hard and/or costly to obtain the input of key customers throughout the life cycle of a product. In addition, as the number of customers to be interviewed is limited, they may not be representative for all types of customers.

In this paper, we investigate whether there is another way of estimating the perceived value of features in a software product using an indirect and objective measure. According to Woodruff [19] customer perceived value is closely related to the usage. Here, our aim is to shed light into how feature usage is related with customer perceived value, and whether this measure can be used to estimate low value features that could be removed from the system in the next releases.

The paper is organized as follows: Section 2 discusses the related work. Section 3 provides details of the case study. Section 4 analyses the results of the case study. Section 5 presents the validity threats. Finally, Section 6 concludes the work.

2 Related Work

2.1 Customer Perceived Value

Software business value is a multidimensional concept. In [9], three dimensions are differentiated: product value, a customer's perceived value and relationship value. Product value is related to the product price and it changes depending on the competitive products. A customer's perceived value is the benefit that customer gains from the product. It can be influenced by customer expectations, previous experiences, cultural background etc. Relationship value is generated from the social relationships between the company and the customer.

Ulaga and Chacour [15] argue that high customer value is a key to creating long-term industrial relationships, which also impact the product value. Authors present an approach to measure and maximize customer perceived value in three steps: (1) analyze how a company understands customer perceived value; (2)

interview the representatives of key customers; (3) suggest a strategy for what to include in a product.

Similarly, in value-based requirements engineering area companies aim to maximize the customer perceived value through selection of the most important requirements for the success-critical stakeholders [20,21,22]. There are two main problems in measuring customers' perceived value: (1) stakeholders may have a different understanding of what value refers to [23]; (2) customer perceived value is a multidimensional concept [15] that makes it difficult to measure.

Ronkko et al. [10] discuss customer perceived value and utility relation in software engineering. They argue that system utility depends on the skills of the users, and therefore customer perceived value is influenced by who uses the system. However, the authors do not provide deeper insights on the nature of the relationship.

Currently, several techniques help to assess customer perceived value [16,17,18]. For example, release planning approaches [16] address this problem by prioritizing requirements through stakeholder surveys. Later the most valuable feature sets are selected and implemented in early releases. The house of quality [18] techniques prioritize customer requirements by mapping them to the design constraints. All of these techniques collect the information through customer surveys. However, it is difficult for participants to precisely estimate customer perceived value because of its multidimensional nature. The wrong evaluation is later on carried on through the whole product development lifecycle. Therefore, there is a need to have objective measures that would guide the assessment process.

2.2 Usage

Existing literature suggests that there are different aspects of system usage [24]: total time, frequency, number of features used etc. Most techniques developed to measure usage focus on the overall system usage, but not of a single feature. Only the recent studies [25] raised the need to explore the system usage in system-centred fashion (i.e. measuring system at the feature level). Sun et al. [26] reports a study about system usage at feature level. However, the data was collected through questionnaires, and therefore represent the subjective opinions of users.

There are two common approaches to monitor how users use the features of a system: (1) extend the software with code that is responsible for monitoring, or (2) design an application that intercepts all events triggered by the observed system when it is used. The main issue using the former method is that the added piece of software increases the complexity of the overall software. In addition, depending on the country where the software is used, the hidden data collection about users activities might violate the privacy laws.

Existing tools that use the second approach (such as aforementioned OpenSpan Desktop Analytics [27] and Google Analytics [28]) overcome afore mentioned limitations, because they do not modify the software that is being monitored. However, such tools are able to show only which applications are running on an operating system or web browser. They do not provide any details related to the feature usage.

Another set of tools such as Microsoft Spy++[29], or the method presented by Atterer et al.[30] provide detailed information on how users use a system by monitoring activities of users, such as mouse clicks and key strokes. However these tools collect too much noise, because they were not created with the aim to identify the usage of the features. For example, such tools catch the events raised by random mouse clicks which do not change the behaviour of the system. Then it becomes difficult to automatically filter out the noise and determine which unique features were executed.

Due to these limitations, we developed a tool that identifies features and provides their usage information when conducting the case study presented in the following section.

3 Case Study

We conducted a case study to explore the relationship between perceived value of features and their usage by customers. Our research question was as follows:

– RQ1 – How does customer perceived value for features relate to their usage?

We selected nextrailer.net web based movie recommender system as the case application developed by a startup company. This startup company, being at an earlier phase in development, agreed to apply our approach for better understanding the customer perceived value for the features of their product. The system contains movie trailers database where users can find their favourite movies by using filters. The system contained 30 features at the time of the case study and had 20 daily users (150 users in total). For the case study we chose 20 features of the nextrailer.net application.

We invited all the system users of the case application to participate in our case study by sending emails. At the end, 19 users accepted to participate in this case study.

3.1 Case Study Design

We used two operational measures to answer RQ1: 1) customer perceived value, and 2) feature usage.

In this study, we first required to define what a feature is before designing any measurement instrument, as our investigation would be at feature level. We adopted the following definition by Eisenbarth et al. [31]: "A feature is an observable unit of behaviour of a system triggered by the user".

As for the customer perceived value, we used Woodruff's definition [19]: "Customer value is a customer's perceived preference for and evaluation of those product attributes, attribute performances, and consequences arising from use that facilitate (or block) achieving the customer's goals and purposes in use situations".

We used 100 point method (also known as 100 dollar method) [13] for measuring the customer perceived value. This method is used in requirements prioritization area to identify the most important requirements for the customers.

Specifically, the customers have to distribute 100 points among all features according to their value. As we were interested in measuring the relative customer perceived value of features, this method fits our needs.

To measure usage of features, we designed a JavaScript library that intercepts all on click events raised by DOM elements that have title attribute. We asked the developers to add this library to the system and to make sure that elements that represent features of the nextrailer.net contain unique title attributes.

We automatically collected the data about the usage of the nextrailer.net for two months. Finally, the following information was sent to our server: the username of a user that triggers the onClick event while using a feature, the timestamp, and the title attribute of the element that represents a feature. For the further analysis we aggregated and extracted data using SQL queries.

4 Analysis and Results

After collecting the data, we analysed the usage of features and perceived value recorded for those features. To explore the relation between customer perceived value and usage of features we considered four scenarios and categorised the features as shown in Fig. 1: (I) a feature is rarely used and not too valuable; (II) a feature is rarely used, but valuable; (III) a feature is frequently used and valuable; (IV) a feature is frequently used, but not too valuable.

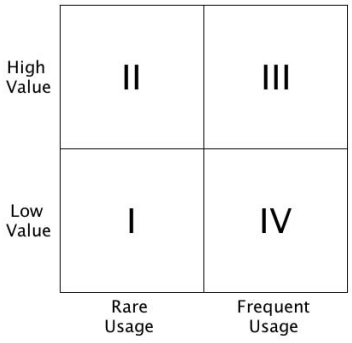

Fig. 1. Feature categorization according to usage/customer perceived value

In order to decide whether a feature is rarely/frequently used and valuable/not too valuable we needed to define threshold values. To do that, for each participant, we computed average usage and average perceived value, and then calculated relative values with respect to average. After this, we categorised the features into one of the four groups. If a feature rating is less than the average rating, then the feature is labelled as 'not too valuable', otherwise as 'valuable'. If a feature is used less than the average usage then the feature is labelled as 'rarely

used', otherwise as 'frequently used'. In this case, the center point in Fig 1 is the crossing point between relative usage and relative customer perceived value for each separate participant.

Fig 2 shows an example of relative feature usage/perceived value diagram for one of the participants.

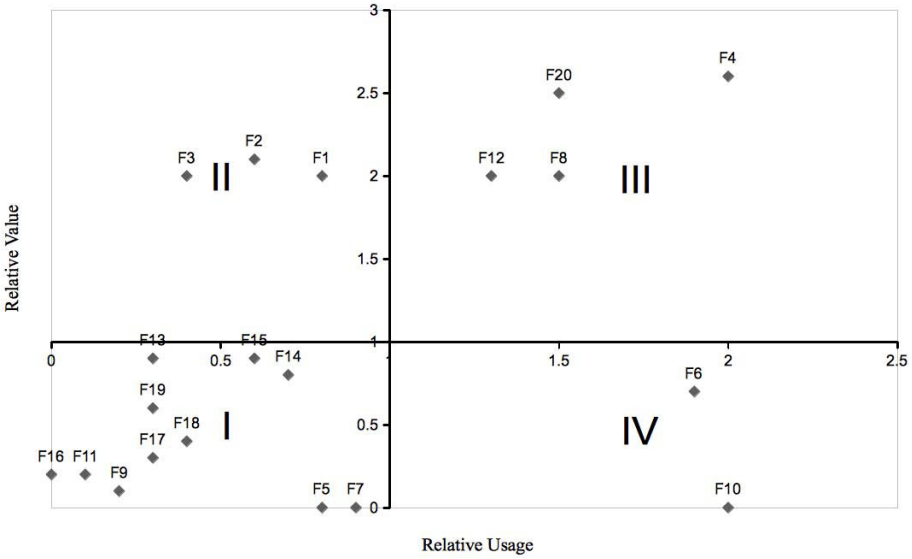

Fig. 2. An example feature usage/perceived value diagram for a participant

In Table 1, Columns I to IV show the features of the case application and their categories (Scenario I, II, III, IV) with respect to the number of users. For example, 7 of the 19 users found to rarely use F1, and they also stated this feature is of low value, whereas 2 of the users found to use the feature frequently and they also think that this feature is of high value.

Scenario I and Scenario III indicate a some correlation between feature usage and perceived value. In total, 65.5% cases indicate a correlation between feature usage and customer perceived value (49% of cases fall in Scenario I and 16.5% of cases fall in Scenario III).

However, for our purposes in this study, it was important to understand Scenario II (when a feature is rarely used, but perceived as valuable), and Scenario IV (a feature is frequently used, but perceived as not too valuable) as one of our aims is to explore when/why usage does not correlate to perceived value. To do that, we interviewed the participants. The results showed that the main five reasons were the following:

Table 1. The categories of features with respect to number of users

Feature	Short description	I	II	III	IV
F1	Watch trailer	7	6	2	4
F2	Store movie	3	8	7	1
F3	Mark seen movies	2	10	5	2
F4	Navigate to the next trailer	6	4	8	1
F5	Navigate to the previous trailer	13	6	0	0
F6	Select trailer using thumbnail list	2	3	10	4
F7	Add movie to the seen list	2	0	8	9
F8	Add movie to the watch list	3	1	11	4
F9	Report a broken trailer	15	4	0	0
F10	**Rate a movie**	**5**	**0**	**2**	**12**
F11	Quick filters	11	6	2	0
F12	Filters	5	11	3	0
F13	Play movie from the seen and watch lists	15	4	0	0
F14	Delete movie from the seen and watch lists	15	4	0	0
F15	See IMDB information about a movie	16	2	1	0
F16	**Register Nextrailer.net**	**17**	**2**	**0**	**0**
F17	**Login Nextrailer.net**	**16**	**3**	**0**	**0**
F18	**Login Social Network**	**14**	**5**	**0**	**0**
F19	**Login Google**	**15**	**4**	**0**	**0**
F20	Search trailer by title	4	11	4	0

R1 Participants reported that some features were very valuable even though they did not need to use them frequently. For example, one stated that 'report a broken trailer' was a high value feature, because the quality of videos could be increased in this way, but there were not a lot of low quality videos in the nextrailer.net website. This reason was reported 66 times in total and for the majority of the features falling in Scenario II. This finding suggests that it would be interesting to explore in the future the relation between discretionary usage [32] and customer perceived value. Discretionary usage considers that some features intended to be used less than others and is defined as follows: $X = A/B$, where X is discretionary usage, A is number of times that specific software functions/applications/systems are used, B is number of times they are intended to be used. The biggest challenge is to estimate B, because the results of this case study shows that different users use features differently.

R2 Participants reported that they made a mistake in distributing the scores among features by giving too high or too low scores. For example, a few participants reported that it was obvious that some features have to be there and they were necessary for minimal site operation, and therefore they rated the value of those features as low. In addition, some participants stated that they mistakenly gave too high scores to some features. This reason was reported for the features falling in Scenario II and Scenario IV. In total, 14 mistakes identified after the interview. This indicates that it is difficult to

reliably evaluate customer perceived value just asking the opinions of users. Therefore, there is a need to find reliable objective measures for measuring or estimating customer perceived value.

R3 Participants reported that some of the features were not valuable for them, but they were part of a process and therefore there was no other way to bypass them. One example given was that when users use the feature 'Mark a see movie', the pop up appears and they had to rate that movie. This in turn increased the usage of 'Rate a movie' feature. This reason was reported 16 times for most features falling in Scenario IV, and it was stated 12 times for 'Rate a movie' feature. This explains most of the cases in Scenario IV. Only 2 participants reported that they like to express their opinion and use the rating system. Usually, such features are known to the company before hand as these dependencies exist due to the architecture of the system. This should be taken into account when making decisions to remove features based on usage information.

R4 Participants reported that the feature was of low value and they believe that usage would decrease in time; that is such features would move to Scenario I from Scenario IV. For example, the feature 'Mark seen movies' was not perceived as very valuable by some users and in the future their profiles would be filled with this information, and therefore the amount of usage is expected to decrease. This reason was reported 9 times for the features falling in Scenario IV. This suggest that, to discover increasing/decreasing patterns of usage, there should be a long period of time monitoring; especially for the systems that users do not use daily. For example, it is not intended that users use movie recommender system every day.

R5 Participants specified that some features were valuable for them because of other reasons than intended by the feature. For example, they stated that 'social network login' did not require to create a new user and remember its credentials, thus some users consider it as an advantage. This reason was reported for 13 times for the features falling in Scenario II. In contrary, some of the participants value nextrailer.net registration and login since they are afraid to give permissions to their personal data that is on social networks. Such cases appear relatively small number of times. Nevertheless, it just confirms that customer perceived value is a multidimensional concept, which needs careful investigation.

The results also showed that in some cases features that are perceived as valuable by the company, tend to have a small customer perceived value, and therefore they might appear in Scenario I or IV whereas they are perceived as valuable by the company. For example, the features F10, F16, F17, F18, and F19 (marked in bold font) were perceived as valuable by the company that developed the case application, because these features are used to collect user personal information: registration details, login, movie ratings. This is an important result of this study as a company would be aware of such features in advance, and can take decisions considering this in addition to customer perceived value/usage.

Table 2. The number of users and their reasons for Scenario II and IV per feature

	Scenario II			Scenario IV		
Feature	R1	R2	R5	R2	R3	R4
F1	6	0	0	3	1	0
F2	8	0	0	0	1	0
F3	10	0	0	1	0	1
F4	1	3	0	0	0	1
F5	3	3	0	0	0	0
F6	0	3	0	2	2	0
F7	0	0	0	2	0	7
F8	1	0	0	3	0	1
F9	4	0	0	0	0	0
F10	**0**	**0**	**0**	**0**	**12**	**0**
F11	3	3	0	0	0	0
F12	11	0	0	0	0	0
F13	4	0	0	0	0	0
F14	4	0	0	0	0	0
F15	2	0	0	0	0	0
F16	**0**	**0**	**2**	**0**	**0**	**0**
F17	**0**	**1**	**2**	**0**	**0**	**0**
F18	**0**	**0**	**5**	**0**	**0**	**0**
F19	**0**	**0**	**4**	**0**	**0**	**0**
F20	11	0	0	0	0	0

In Table 2, we present the number of participants reported the respective reasons (listed above) for why/when usage and perceived value for the same feature did not correlate (that is Scenario II and IV). For example, all 6 users, who mentioned that F1 is a high value feature while they rarely use it (Scenario II), stated R1 as the main reason.

The results show that for the features falling in Scenario IV (high usage/low value), R3 is the dominating reason. This indicates the dependency between features in the architecture. For Scenario II (low usage/high value), the dominating reason is R1. This states that it should be important to consider whether a feature is intended to be used frequently or not when making a value-based decision.

5 Threats to Validity

We discuss the validity threats of this study according to categorization suggested by Runeson and Host [33]: construct validity, internal validity, external validity and reliability.

Construct Validity. Construct validity refers to what extent the operational measures represent what is investigated according to the research questions.

We use two operational measures: customer perceived value and feature usage. One validity threat could be that automatic measurement of the feature usage is incorrect. To mitigate this threat we asked developers to insert unique title attributes on the DOM elements that represent features. Then we tested if this information is really collected by using all the features.

Another threat was misunderstanding of users what perceived value means while they are distributing 100 points to features. To mitigate this threat, we provided the definition of 'customer perceived value' by Woodruff [19] and had a discussion about it so that every participant has similar understanding about it. In addition, we made our analysis per user and tried to capture the relative perceived value and how it relates to its usage. We believe that our conclusions should not be significantly affected by this threat.

Internal Validity. Internal validity concerns the causality relations. One validity threat could be overlooking some factors, which could have affected the outcome and that we did not control. Another validity threat could be that participants were not familiar with the system and used it for a small amount of time. To avoid this threat we selected only the existing participants of the system and measure the feature usage for 2 months. The other validity threat could be participants rating features to satisfy the final objective of this study. To mitigate this threat, the participants were not informed about the main objectives of the research study at the beginning.

External Validity. External validity refers to what extent it is possible to generalize the findings to different or similar contexts. One validity threat could have been the small number of participants in this study. Moreover, as we conducted only one case study using a web application, we do not know how much the results are generalizable to other type of applications. However, here our intention was not to seek for generalizable results, but rather understand the factors that could affect the relationship between usage and perceived value and generate hypothesis for future work.

Reliability. Reliability reflects to what extent the data and the analysis depend on the specific researchers. One threat could be misinterpretation of the answers that we collected during the interview to identify reasons for when/why usage and perceived value do not correlate. To mitigate this threat, we validated with each participant how we interpreted their answers.

6 Conclusions and Future Work

In this paper, we explored the relation between usage of features and their perceived value by the users. The results of this study confirmed that it is very difficult to reliably measure customer perceived value through questionnaires asking the opinion of users, as they might have some implicit assumptions about what value means. Measuring usage of the features helps in understanding these

assumptions as well as identifying mistakes made by the customers when evaluating value of features.

Overall, we conclude that usage of features considering the aforementioned factors is a promising approach to indirectly estimate the customer value of features. However, it is important to incorporate into usage measure definition whether a feature is intended to be used frequently or not. For example, this can be done through using discretionary usage [32] as a metric. We also identified that it is important to monitor a system for a longer period of time to observe increasing/decreasing patterns of usage as some features might be from the beginning intended to be used frequently, whereas others rare.

As future work, we plan to test these hypothesis and conduct more exploratory case studies for different types of applications to investigate whether there are more reasons why/when feature usage does not correlate to its value perceived by customers.

References

1. Boehm, B.: Value-based software engineering. In: Software Engineering Notes (2003)
2. Mohamed, S.I., Wahba, A.M.: Value estimation for software product management. In: Industrial Engineering and Engineering Management (2008)
3. Vahaniitty, J.: Key decisions in strategic new product development for small software product businesses. In: Euromicro Conference (2003)
4. Berry, M., Aurum, A.: Measurement and decision making. In: Value-Based Software Engineering (2006)
5. Egyed, A., Biffl, S., Heindl, M., Grünbacher, P.: A value-based approach for understanding cost-benefit trade-offs during automated software traceability. In: Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering (2005)
6. Boehm, B.: Value-based software engineering: reinventing. In: ACM SIGSOFT Software Engineering Notes (2003)
7. Bauer, H.H., Hammerschmidt, M., Braehler, M.: The customer lifetime value concept and its contribution to corporate valuation. In: Yearbook of Marketing and Consumer Research (2003)
8. Park, Y., Park, G.: A new method for technology valuation in monetary value: procedure and application. In: Technovation (2004)
9. Barney, S., Aurum, A., Wohlin, C.: A product management challenge: Creating software product value through requirements selection. Journal of Systems Architecture (2008)
10. Rönkkö, M., Frühwirth, C., Biffl, S.: Integrating value and utility concepts into a value decomposition model for value-based software engineering. In: Bomarius, F., Oivo, M., Jaring, P., Abrahamsson, P. (eds.) PROFES 2009. LNBIP, vol. 32, pp. 362–374. Springer, Heidelberg (2009)
11. Bullock, J.: Calculating the value of testing: How to present testing as a business process investment. Software Testing and Quality Engineering (2000)
12. Saaty, T.L., Vargas, L.G.: Models, methods, concepts and applications of the analytic hierarchy process. Springer (2001)

13. Ahmad, A., Shahzad, A., Padmanabhuni, V.K., Mansoor, A., Joseph, S., Arshad, Z.: Requirements prioritization with respect to Geographically Distributed Stakeholders. In: Computer Science and Automation Engineering (2011)
14. Karlsson, L., Berander, P., Regnell, B., Wohlin, C.: Requirements prioritisation: an experiment on exhaustive pair-wise comparisons versus planning game partitioning. In: Empirical Assessment in Software Engineering (2004)
15. Ulaga, W., Chacour, S.: Measuring customer-perceived value in business markets: a prerequisite for marketing strategy development and implementation. In: Industrial Marketing Management (2001)
16. Greer, D., Ruhe, G.: Software release planning: An evolutionary and iterative approach. In: Information and Software Technology (2004)
17. Aurum, A., Wohlin, C.: Aligning requirements with business objectives: A framework for requirements engineering decisions. In: Proceedings of Requirements Engineering Decision Support Workshop (2005)
18. Park, T., Kim, K.J.: Determination of an optimal set of design requirements using house of quality. Journal of Operations Management (1998)
19. Woodruff, R.B.: Customer value: the next source for competitive advantage. Journal of the Academy of Marketing Science (1997)
20. Aurum, A., Wohlin, C.: A value-based approach in requirements engineering: Explaining some of the fundamental concepts. In: Sawyer, P., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 109–115. Springer, Heidelberg (2007)
21. Wohlin, C., Aurum, A.: What is important when deciding to include a software requirement in a project or release? In: International Symposium on Empirical Software Engineering (2005)
22. Wohlin, C., Aurum, A.: Criteria for selecting software requirements to create product value: An industrial empirical study. In: Value-Based Software Engineering (2006)
23. Kortge, G.D., Okonkwo, P.A.: Perceived value approach to pricing. In: Industrial Marketing Management (1993)
24. Doll, W.J., Torkzadeh, G.: Developing a multidimensional measure of system-use in an organizational context. Information and Management (1998)
25. Burton-Jones, A., Gallivan, M.J.: Toward a deeper understanding of system usage in organizations: A multilevel perspective. Mis Quarterly (2007)
26. Sun, H.: Understanding User Revisions When Using Information System Features: Adaptive System Use and Triggers. MIS Quarterly-Management Information Systems (2012)
27. OpenSpan Desktop Analytics, `http://www.openspan.com/products/desktop_analytics` (last visited on November 27, 2012)
28. Google Analytics, `http://www.google.com/analytics` (last visited on November 27, 2012)
29. Microsoft Spy++, `http://msdn.microsoft.com/en-us/library/aa232254v=vs.60.aspx` (last visited on November 27, 2012)
30. Atterer, R., Wnuk, M., Schmidt, A.: Knowing the users every move: user activity tracking for website usability evaluation and implicit interaction. In: Proceedings of the International Conference on World WideWeb (2006)
31. Eisenbarth, T., Koschke, R., Simon, D.: Locating Features in Source Code. IEEE Computer (2003)
32. ISO/IEC TR 9126-4, Software engineering – Product quality – Part 4: Quality in use metrics (2004)
33. Runeson, P., Host, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering (2009)

Business Incubation Practices and Software Start-up Success in Turkey

Gozem Guceri-Ucar and Stefan Koch

Department of Management, Bogazici University, 34342 Bebek, Istanbul, Turkey
gozem.guceri@gmail.com
stefan.koch@boun.edu.tr

Abstract. This study presents preliminary findings of a longitudinal qualitative study concerning business incubators in Turkey, and their effectiveness in enhancing the success and sustainability of new software ventures. A field study was conducted to gain an in-depth understanding of incubation experiences and software start-up success. Results were combined with literature review findings to derive conclusions relating to software start-up success and the value-adding contribution of business incubators in Turkey.

Keywords: Entrepreneurship, Business incubator, Start-up success, Software business.

1 Introduction

The start-up period is when a business is most vulnerable [1], and many start-ups seek funding as well as other resources during this time. Business incubators support firms during this critical and risky timeframe, helping them survive and grow by providing resources and facilities as well as guidance and consultancy.

This article presents first findings of a longitudinal qualitative study concerning business incubators in Turkey, and their effectiveness in enhancing the success and sustainability of new software ventures. Our aim is to explore the concept of business incubation both in literature and in practice, and identify emergent constructs that may relate to tenant success.

The initiation of the business incubation idea dates back to the 1950s, and today the number of business incubators throughout the world is estimated to be between 4000 and 5000. In Turkey, this number is approximated around 40-50 [2], but it is rapidly increasing.

Voisey et al. [3] define business incubation as both a place and a process – simultaneously. It is a combination of business development processes, facilities and people, uniquely combined to support and grow new ventures by supporting them through the early stages of their foundation. Hackett and Dilts define an incubator as "...a shared office-space facility that seeks to provide its incubatees with a strategic, value-adding intervention system of monitoring and business assistance" [4, p. 57].

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 178–182, 2013.

While all business incubators share the common goal of nurturing newborn companies, it is possible to further classify them according to: (i) main purpose of the incubator, and (ii) funding scheme (sponsor) of the incubator [5].

Hackett and Dilts [4] observe that while much attention has been devoted to the description of incubator facilities within past studies, less attention has been focused on the tenants; what they are trying to achieve, what innovations they seek to develop and diffuse. They also identify that there is a lack of studies concerned with the incubation outcomes that have been achieved. We aim to fill these gaps by focusing on the tenants, their experiences throughout the incubation process, and their lives after graduation.

In order to determine whether a set of constructs relate to both start-up and incubator success, one has to first set a solid definition of success and underlying critical success factors (CSFs).

The definition of success depends heavily on the incubator's purpose of establishment. Knowing this purpose also enables us to identify CSFs to monitor progress with respect to goals and objectives. Several CSFs have been mentioned in literature, including flexibility to adjust to circumstances [6], strength and organization of tenants' planning activities [7,8,9], monitoring and continuous feedback of incubator management [8,9], organizational learning [10], the participation of financers in tenant capitalization [11], and proper assessment of incubator-tenant fit [12]. Hackett and Dilts [4] summarize incubator variables associated with incubatee success as follows: (i) incubatee selection processes, (ii) internal incubator network formation, (iii) incubator-industry network and incubator-support services network density, (iv) incubator manager-tenant relationships, (v) incubator effectiveness, (vi) level of incubator development, (vii) procedural standardization and policy formalization.

2 Exploring Business Incubation and Software Start-up Success: A Field Study

Our participants included owners and managers of 12 start-ups, each positioned in one of the 5 business incubators listed in Table 1. 9 of the 12 incubator tenants were developing software as their core business for different industries including software as a service in telecommunications (4), sports analytics (1), health services (1), and information technologies (3). Among the remaining companies, 2 were developing electronic equipment and 1 was a textile and accessories e-commerce business. Table 1 summarizes information regarding the incubators in which our participants' companies are currently located. We also interviewed 3 software companies that had chosen not to start their business in an incubator.

Our study consisted of semi-structured interviews that lasted 45-60 minutes. 9 incubator tenants were still in incubation at the time of their interviews, while 3 had graduated, and 1 of these was no longer in operation. The oldest among the 15

Table 1. Incubators included in the field study

Incubator	Type	Incubator location	Age	# of tenants	# of tenants interviewed	Management interview
Turk Telekom	CPI	Company R&D center in ITU technology park	< 1 year	2	2	Yes
AveaLabs	CPI	Company R&D headquarters	2 years	5	2	Yes
KOSGEB Bogazici TEKMER	BIC	Technology park on Bogazici University campus	>10 years	23	5	No
BU Hayat Et	UBI	Bogazici University campus	< 1 year	4	1	Yes
ITU ARI Cekirdek	UBI	Technology park on ITU* campus	>10 years	29**	2	No

CPI: Corporate Private Incubator – BIC: Business Innovation Center – UBI: University Business Incubator

* Abbreviation for Istanbul Technical University

** Approximate number as of July 2012

companies was founded in 2008, while the youngest 2 were founded in August and September 2012. Interviews were also held with managers of 3 of the incubators involved, and some follow-up information was requested from both incubator management and incubatees on how they plan, monitor and document their progress.

The definition of incubator success depends on the purpose of the business incubator, while tenant success is defined by the purpose and objectives of each company. When asked their primary objectives and definition of tenant success, the managers of 3 incubators responded as shown in Table 2 below:

Table 2. Incubators' primary objectives and definitions of tenant success

Incubator	Primary Objective	Secondary Objective	Definition of tenant success
Turk Telekom	Contribute to the economy by supporting young firms	Strengthen university partnerships, support corporate image	Tenant survival until graduation
AveaLabs	Create potential suppliers for Avea and its group companies	Technology transfer, spin-offs	Creation of marketable and value-generating products by graduation
BU Hayat Et	Contribute to the economy by supporting young firms	Innovation and technology transfer	Tenant survival until graduation, creation of marketable products by graduation

The items used by interview participants in defining pre-graduation success, and the CSFs associated with each item have been tabulated below in Table 3:

Table 3. Success definitions and related critical success factors of interviewed firms

	Success Categories						
	Product /service development	Cash flow		Clients & Networking	Strategic Partnerships	Human capital	International-ization
		Funding	Sales				
Critical Success Factors	Product development	Additional government funding	Sales opportunities for demo versions	Gaining access to clients	Synergy with other start-ups in the same facility	Employee growth	Oversees subsidiary initiative
	Product commercialization	Sponsors and venture capital	Sales of secondary products or services	Creating a client network	Development of a supplier network	Employee competence level	International clients
	Alignment of business plan and development plan	Bank loans	Sales of commercial product	Client feedback	Partner feedback	Employee competence development / training	Representation in international industry events

The success categories listed in Table 3 were determined by consolidating success definitions provided in literature and the success indicators identified by the start-ups within their business plans, which were prepared and submitted during the incubator's tenant application process. We also asked the same questions to the companies which had not benefited from incubation. Their goals for the initial 2 years of their company were similar to those tabulated above, and did not include any additional items. There was also no difference between the pre-graduation goals and CSFs of companies specializing in software and other areas.

The categories in Table 4 were derived from literature and cross-checked with the classifications used by incubator managers during our interviews.

Table 4. Factors that contribute to tenant success

Facilities	Services	Management
Location (proximity to university, technopark, target market, access to qualified workforce)	Start-up training	Founder (sponsor) characteristics
	Business consulting	Tenant selection criteria
Open/closed office	Financial consulting	Tenant – incubator fit
	Market knowledge and intelligence	Mentorship and monitoring of tenant progress
Work environment (workspaces, meeting rooms, etc)	Access to financing (e.g. links with business angels)	Incubator business model (non-profit vs. for-profit)
	Social services	Incubation period
	Technical services	Organizational learning (Age of incubator, experience of incubator management)

3 Conclusions and Future Research

The following propositions were derived regarding how business incubator practices may contribute to tenant success:

(a) Location of the incubator will contribute to tenant success if it is in close proximity to an institution conducting innovative research in a similar area, if it provides locational advantage in terms of access to target customers, or if it provides ease of access to qualified employees.

(b) An open office structure will contribute to tenant success by enabling networking among tenants and creating opportunities for strategic partnerships.

(c) Quality of the work environment provided by the incubator will enhance tenants' efficiency in product development, and contribute to human capital by increasing employee motivation.

(d) Quality and scope of services offered by the incubator will contribute to tenants' cash flow management and networking activities, development and refinement of their business plan, and the development of their human capital.

(e) The criteria for tenant selection and assessment of incubator-tenant fit will affect multiple dimensions of tenant success, such as product development, networking with potential clients and partners, and funding.

(f) Mentorship provided by incubator management will contribute to tenants' planning, networking, and cash flow management.

(g) The age of the incubator and the experience level of incubator management will affect multiple dimensions of tenant success, such as networking with potential clients and partners, cash flow, and human capital.

We will continue with a follow-up study aiming to triangulate our findings. The categories and items within Tables 3 and 4 will be tested through surveys and factor analyses of the results. A complete list of variables and hypothesized relationships will be constructed, following the reliability and validity checks on the identified categories and variables. Surveys will be sent to tenants and graduates of technology incubators in Turkey at different points in time, in order to test the hypotheses through multivariate analysis methods.

References

1. Aernoudt, R.: Incubators: Tool for Entrepreneurship? Small Bus. Econ. 23, 127–135 (2004)
2. Bayhan, D., Ozdemir, A.H.: Technology Based Entrepreneurship and Incubation in Turkey. In: Models for National Technology and Innovation Capacity Development in Turkey, pp. 251–310. Technology Development Foundation of Turkey (TTGV), Ankara (2009)
3. Voisey, P., Gornall, L., Jones, P., Thomas, B.: The measurement of success in a business incubation project. Small Bus. Enterprise Dev. 13, 454–468 (2006)
4. Hackett, S.M., Dilts, D.M.: A Systematic Review of Business Incubation Research. Technol. Transfer 29, 55–82 (2004)
5. Akcomak, S.: Incubators as tool for entrepreneurship promotion in developing countries. In: Naude, W., Szirmai, E., Goedhuys, M. (eds.) Entrepreneurship, Innovation and Economic Development, pp. 228–264. Oxford University Press, Oxford (2011)
6. Scherer, A., McDonald, D.W.: A model for the development of small high-technology businesses based on case studies from an incubator. Prod. Innovat. Manag. 5, 282–295 (1988)
7. Rothaermel, F.T., Thursby, M.: Incubator firm failure or graduation? The role of university linkages. Res. Policy 34, 1076–1090 (2005)
8. Stuart, R., Abetti, P.A.: Start-up ventures: towards the prediction of initial success. Bus. Venturing 2, 215–230 (1987)
9. Fry, F.L.: The role of incubators in small business planning. Am. J. Small Bus. 12, 51–61 (1987)
10. Allen, D.N., McCluskey, R.: Structure, Policy Services and Performance in Business Incubator Industry. Entrep. Theory Pract. 15, 61–77 (1990)
11. Campbell, C., Allen, D.N.: The Small Business Incubator Industry: Micro-level Economic Development. Econ. Dev. Q. 1, 178–191 (1987)
12. Rice, M.P.: Co-Production of Business Assistance in Business Incubators: An Exploratory Study. Bus. Venturing 17, 163–187 (2002)

Ecosystem Health of Cloud PaaS Providers

Garm Lucassen, Kevin van Rooij, and Slinger Jansen

Department of Information and Computing Sciences, Utrecht University,
Princetonplein 5, 3584 CC Utrecht The Netherlands
{g.g.lucassen,k.a.vanrooij}@students.uu.nl, slinger.jansen@uu.nl

Abstract. Customers of Platform as a Service providers are unable to evaluate the risk of their provider going bankrupt. Lacking this information, businesses are effectively putting their critical business services in jeopardy. In this paper, we present a method to evaluate the ecosystem health of eight different PaaS providers. The results of our research enable businesses and individuals to make an informed decision on what PaaS providers to do business with. Additionally, the PaaS providers themselves are given insight into the current state of their ecosystem compared to competitors.

Keywords: Software Ecosystems, Cloud PaaS, Ecosystem Health, Open Source, Repository Mining, Platform as a Service.

1 Introduction

Since 2009, academic and business interest in the Platform as a Service (PaaS) industry has grown exponentially. Yearly Google Scholar hits for the keyword "Platform as a Service" has grown from 83 in 2008 to 3320 in 2012.[1] Gartner expects enterprise public cloud services spending to reach $207 billion by 2016 [1]. Due to this surge in attention, the United States National Institute of Standards and Technology added a definition of PaaS in 2011:

> *"The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."* [2]

Concretely put, PaaS providers host virtual systems that run application stacks facilitating easy deployment of a web-enabled program in a specific programming language, significantly reducing development time and effort for the customer. Key advantages include: dramatically lowering the cost of entry, almost immediate access to hardware resources, lowering IT barriers to innovation, flexibly

[1] http://scholar.google.com/scholar?q="platform+as+a+service"

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 183–194, 2013.

scalable services to meet client demand and enabling new classes of applications and services [3]. Despite this explosive growth in both academic and business relevance, no publicly accessible information is available on the ecosystems health of public cloud PaaS providers. Ecosystem health is defined as *"long-term financial well-being of the business ecosystem and the long-term strength of the network"* [4]. Access to knowledge regarding ecosystem health is crucial for businesses researching the possibility to move their software to the cloud for two reasons [5]. To begin, businesses attempt to seek partners with a robust business ecosystem [6]. Second, the availability of their critical business applications relies on the robustness of their PaaS Provider. Unexpected bankruptcy of a PaaS provider can lead to loss of data or catastrophic downtime. Indications of these kinds of developments are preferably known beforehand.

Because PaaS providers are primarily software businesses, this paper leverages knowledge on the academic field of software ecosystems to study their ecosystem health. Several authors have defined the term software ecosystem [7,8], but this research applies the following definition by Jansen et al., as it builds upon earlier definitions and further abstracts the concept [9]: *"a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts."* A high profile example of a strong software ecosystem is iOS. Apple benefits from the popular App Store, due to which the iOS ecosystem remains the market leader in terms of app availability, app sales and profit margins [10].

In the context of PaaS ecosystem health, the contributors to the long-term strength of the ecosystem are the direct users of the PaaS technology: developers. If a group of developers is actively contributing to the development of PaaS or its extensions, the PaaS itself is more likely to succeed in the long run. This paper uses data of the open source code hosting service GitHub to analyze the ecosystem health of eight different PaaS providers. First, metadata of all software projects or *repositories* that contribute to the ecosystems of each PaaS providers is collected. Next, specific data fields are aggragated into ecosystem health results based on open source ecosystem health measures introduced by Crowston et al. [11]. Finally, statistical analysis extracts relevant insights on the current growth expectations of a PaaS and expected health of their ecosystem.

With the results of our research, businesses will be able to make a more informed decision in choosing a cloud hosting PaaS provider. Moreover, the PaaS providers are presented with a method to gain insight into the state of their ecosystem. This can help them achieve their goals, make better use of available resources, reduce risks, increase revenues [12] and/or compare their business to competitors.

The next section details the research approach. Subsequently, section 3 explicates the data collection methods, while section 4 introduces all indicators which we use to extract insights from the collected data. Section 5 and 6 present and analyze the results themselves. These findings are discussed in section 7. The paper finishes with a conclusion and future research possibilities in section 8.

2 Research Approach

The PaaS providers included in this research are: Azure, Cloud Foundry, dotCloud, Engine Yard, Google App Engine, Heroku, Nodejitsu and OpenShift. This list was created based on two criteria: (1) The PaaS should support easy deployment with one or more development frameworks such as PHP, .Net or Ruby. (2) The solution has to be mature and actively used. PaaS providers that adhere to this criterion have public customers and their share of PaaS modules on the Open Source code hosting service is larger than 1% of all repositories. Four PaaS that meet the first, but not the second criterion are omitted from this research: AppHarbor, AppFog, Amazon Elastic Beanstalk and Nodester.

Relevant comparative measures are necessary to evaluate the health of a PaaS ecosystem. In 2006, a multitude of indicators were collected by Crowston et al. to measure the health of open source ecosystems [11]. We consider four indicators from this research based on three earlier publications [13,14,15] to be appropriate in the context of PaaS ecosystem health:

1. Number of active developers
2. Spin offs
3. Interest in the project
4. Download count

Due to the focus on open source development of large projects, the other indicators collected by Crowston et al. are unsuitable in the context of PaaS or require unobtainable data. For example, the perceived ease of use is not applicable to projects that require integration with another piece of software and code quality can not be measured for ten-thousands of projects accross more than a hundred programming languages. Instead, the selected indicators measure interest in the PaaS Providers of their users, the developers. If developers create applications for a PaaS, its ecosystem will flourish. On the other hand, if developers are not interested in developing for that platform, this indicates that the ecosystem is unhealthy. This paper is based on the assumption that developer interest has a direct correlation to the customer size of the PaaS provider and subsequently the long term health of the ecosystem that belongs to it.

Data was collected from GitHub with a Ruby program developed by the authors of this research paper. Once collected, the data was prepared for analysis by validating completeness, correcting errors and redundancy, transforming to a uniform format and storing it in a database. Statistical analysis was subsequently conducted with the R programming language for statistical computing[2]. Alternative open source code hosting services to GitHub with a publicly accessible API such as Bitbucket, Tigris and Launchpad were excluded. These alternatives host orders of magnitude less repositories than GitHub and their data is less clean. A manual search for the largest PaaS, Heroku, returned 30,748 results on GitHub, compared to at most 247 repositories at the alternatives. Moreover, their databases contain many redundant records and significantly less rich metadata.

[2] `http://www.r-project.org/`

3 Data Gathering

Multiple methods to extract data from GitHub are available, including the official API or screen scraping. For this research, a small Ruby application leveraging the octokit gem[3] was written that conducts repository searches through keywords of PaaS provider names. Because developers use a multitude of different names to refer to the same PaaS, multiple searches with variations of each name were conducted. For instance, while Heroku only requires 'Heroku', Google App Engine has repositories with names as: 'GAE', 'GoogleAppEngine', 'Google App Engine' or just 'App Engine'. Additionally, GitHub hides private repositories from search results, restricting data collection to public repositories. The data collection code can be found on GitHub[4]. All data was collected on January 4, 2013.

3.1 Key Data Elements

The Ruby application collects the following data elements for every repository:

1. Created at date
2. Description
3. Number of followers
4. Fork boolean
5. Number of forks
6. Programming Language
7. Name of repository
8. Owner
9. Private boolean
10. Push date (latest update)
11. Size
12. Type

Unfortunately, users are unable to retrieve the number of downloads of a repository with the API due to technical limitations[5]. On top of that, GitHub does not display any download count data on the website, ruling out the possibility of screen scraping. As a result, it is impossible to assess PaaS ecosystem health with the download count indicator of Crowston et al [11].

3.2 Data Preparation

Some repositories contain two or more different keywords referring to a single PaaS in their description or name fields. As a result, multiple returned results of the same repository for different keywords create redundant records in our data set. On the other hand, 574 repositories support multiple PaaS platforms, necessitating some redundant records. For example, if a repository supports both Heroku and Google App Engine a second entry of the record is appropriate. Instead of leveraging the unique GitHub id assigned to every repository, a unique primary key was composed by combining the PaaS platform keyword, username of the repository owner and the repository name itself. GitHub enforces a strict

[3] http://rubygems.org/gems/octokit
[4] http://github.org/gglucass/seco
[5] http://stackoverflow.com/questions/6198194/how-to-see-count-of-project-downloads-on-github

unique naming protocol which ensures that no false duplication positives are raised. Next, we removed all repositories with a size of 0 from the data set. We consider these repositories as false positives because they are empty and thus do not indicate a concretized developer interest.

Finally, a 1% sample of all repositories was used to determine the amount of remaining false positives in the data-set. In this sample, a repository is considered a false positive when the project itself is not directly related to that specific PaaS. Examples are repositories that mention the PaaS, but in a different context, e.g. 'this repository provides an alternative to PaaS X'. The sample was gathered by performing a SQL query on the data-set which randomly selected 1% of all repositories. All repositories from this sample were subsequently manually verified. Although a random sample has its limitations, the sample gives an overall indication for the total data-set. We believe that this sample is representative for the total data-set and therefore should be taken in consideration when reviewing our results.

The sample resulted in 29 (5.8%) false positives, of which the majority are from Google App Engine (13) and Heroku (9). CloudFoundry, dotCloud and Nodejitsu had no false positives. However, due to their smaller sample size a smaller number of false positives is expected. These results do not indicate that repositories of these PaaS providers are free of false positives. Based on the low percentage of false positives found in the sample, we are confident that the collected data-set provides an accurate representation of the contemporary ecosystems of the selected PaaS providers.

4 Indicators

This section explicates three indicators and what types of sub-indicators each is comprised of. Each sub-indicator is accompanied by a short explanation of how it is calculated.

4.1 Active Developers

Because the development of a project first and foremost relies on voluntary contributions of developers, Crowston et al. state that one indicator of success is the absolute number of developers involved in an open source project [11]. By looking at the number of active developers in the past year as well as per week, a more balanced representation is generated.

Active Developers in the Past Year. The total number of active developers developing on top of a PaaS provides a direct measure of developer interest in the past year. Crowston et al. propose to measure this by collecting the number of developers who are **formally** associated with a project. To adhere to this requirement, we calculate this measure as the sum of all unique owners of repositories updated in the past year.

Active Developers of Unique Repositories in the Past Year. Not all repositories are created equal. Many are copies of original projects or slight derivates. This sub-indicator only takes into account non-fork repositories, i.e. unique projects that were started from scratch instead of based on another project. Likewise, this sub-indicator is calculated as the sum of all unique owners of unique repositories updated in the past year.

Active Developers Per Segment of Time. The week to week variety in number of actively contributing developers measures group activity in time between releases or *cycle time* [11]. For each week in the past year, the sum of all unique owners of repositories is calculated. These numbers are subsequently divided by the total number of developers in the past year and in turn presented in a line-graph. With this graph, the relative weekly activity of developers for the PaaS is visualized.

4.2 Spin Offs

Crowston et al. mention spin offs as an element of recognition, which in turn is a measure for project success [11]. A spin off is a derivative of a previous projects or a new project entirely. In the context of this research, the two simplest measures of spin offs are the number of forks (derivative of previous project) and projects (repositories). However, on GitHub a fork is also considered a repository. To not let this skew the results, a separate sub-indicator representing the number of repositories which are not forks is included.

Total Repositories. Repositories are a direct indicator of the number of spin offs for a PaaS. The total number of repositories which contain the PaaS keyword is the most basic method to measure the contribution of spin offs to project success.

Unique Repositories. Popular repositories on GitHub have many forks, which are often direct copies of the original repository. Multi-platform repositories that are exceptionally popular increase the total number of repositories for all PaaS providers, although one or more of these might not be that popular at all. To discount this phenomenon, this sub-indicator is calculated as the total number of original repositories.

Forks. In theory, total repositories includes forks, but in practice this data is incomplete because of two factors. First, GitHub provides private repositories to premium members; effectively shielding us from accessing that data. Second, some repositories are forked but deleted sometime later. Luckily, for each repository GitHub counts the number of forks made regardless of these restrictions. For each PaaS the sum of all forks is taken to include potentially lost data.

4.3 Interest

General interest in a project is different from the first two indicators, which are focused on the quantity of developers and spin-offs. Instead, this indicator

focuses on *level of activity*. Crowston et al. recommend to examine development logs for evidence of software being written and released. In absence of this data, we measure interest as the combination of passive interest, interest longevity, number of unique programming languages and multi-platform projects.

Total Number of Followers. Followers are GitHub members who have starred a repository, a mechanism that is used to be kept up-to-date with changes. A follower does not actively contribute to development. Thus a follower is an indicator of passive interest in the project. Calculated as the sum of all follower counts for each repository of a PaaS.

Number of Unique Programming Languages. The diversity of programming languages that are used to develop for a PaaS indicate how broad of an interest there is in leveraging the strength of that PaaS. The metric is the total number of unique languages used for each PaaS.

Number of Multi-platform Repositories the PaaS is a Part of. Some popular repositories support more than one PaaS. Counting all repository name + owner combinations which also occur for other PaaS providers measures the interest of high profile development projects.

Number of Repositories Updated at Least Once. Many repositories and forks of repositories are created but never updated. Counting the number of repositories that are updated at least once after creation provides an indication of the interest longevity of developers in the PaaS.

Created at Date Smaller Than Push Date / Created at Date >= Than Push Date. Many repositories are forked or created and subsequently never updated, effectively rendering these projects as dead. Calculating the proportion of these repositories in relation to repositories that are updated, indicates the longevity of developer interest in developing on top of a PaaS. This ratio is calculated by dividing the number of repositories that were updated after creation by the number of repositories that have the same updated as created date.

5 Results

The data collection program yielded a dataset of 55,927 repositories, 35% (19,859) of these are unique repositories, i.e. repositories that are not forked from another repository on GitHub. In total, 50,057 forks were made, of which 85% are forked from a unique repository. In the last year 24,987 developers have contributed to any of these repositories. The average PaaS provider has 6991 repositories, 2482 unique repositories and 4945 developers. On average, a repository has 0.895 forks and 4.579 followers. Table 1 summarizes these findings. Only taking in account unique repositories, these numbers more than double to 2.16 forks and 10.8 followers. Compare this to the numbers of strictly forked repositories, which on average has 0.1983 forks

Table 1. Descriptive statistics

Number of ...	Minimum	Median	Mean	Maximum
Repositories per PaaS	924	2785	6991	29,980
Unique repositories per PaaS	318	720.5	2482	10,400
Developers per PaaS	618	1948	4945	20,320
Forks of repositories	0	0	0.895	1212
Followers per repository	0	1	4.579	7567

and 1.154 followers. The difference in popularity is close to a factor of ten for both values, indicating that a spin-off of a project rarely is an interesting new project itself. Instead, most forks are likely duplicates of the original project, which developers further develop to contribute to the expansion of that project.

5.1 Indicator Results

The raw results of the indicators presented in section 2 are presented in Table 2 and Figure 1. Aside from the updated vs. non-updated repository, all metrics confirm the expectation that Heroku and dotCloud are the most and least popular PaaS providers, respectively. Furthermore, the total number of forks exceeded the number of repositories minus the number of unique repositories for every single PaaS. This validates our expectation that data is lost due to private and deleted repositories, as well as the necessity of inclusion of this sub-indicator. The next section further analyzes these results.

Table 2. Indicator results

	a	b	c	d	e	f	g	h	i	j
Azure	2298	681	3607	1251	4187	19,455	29	1327	27	0.65
CloudFoundry	1244	316	2857	684	2762	11,820	18	1410	54	0.55
dotCloud	496	193	924	418	839	6175	12	328	44	0.69
Engine Yard	1124	193	2713	320	2141	10,805	13	1079	44	0.56
Google App Engine	4660	2243	12,319	5709	7538	39,967	32	4594	87	0.70
Heroku	15,384	5633	29,982	10,402	29,137	148,766	38	10619	228	0.62
Nodejitsu	957	214	1446	318	1596	12,766	7	601	38	0.5
OpenShift	1327	498	2079	757	1857	6361	18	742	51	0.72

Legenda

a Active developers in the past year	f Followers
b Active developers of unique repos	g Unique programming languages
c Number of repositories	h Repositories updated at least once
d Number of unique repositories	i Multi-platform repositories
e Number of forks	j Updated vs. Non-updated ratio

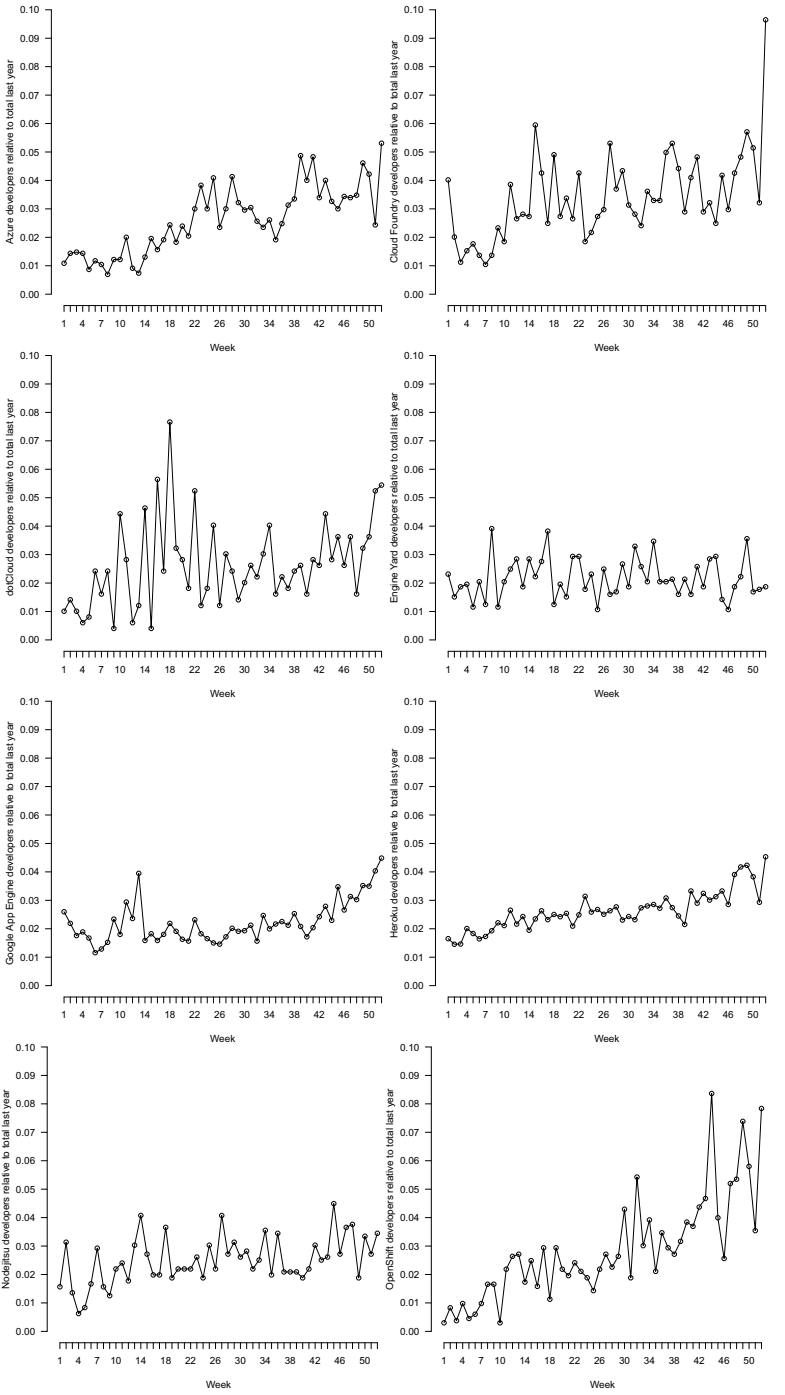

Fig. 1. Graphs displaying relative number of active developers per week

6 Analysis

In this section, the results are further analyzed by looking at the distribution of all indicator metrics and of actively contributing developers in the past year. Table 2 demonstrates Heroku its dominance in the PaaS industry, with the best results for 9 out of 10 sub-indicators. Google App Engine and Azure are in similar comfortable positions, with second place for all sub-indicators and third place for 8 out of 10 sub-indicators, respectively. Although it should be noted that Heroku accounts for more than half of all results for 7 out of 10 sub-indicators. Furthermore, development for Azure includes the least amount of unique programming languages. dotCloud and Nodejitsu are at the bottom of the results, together they have collected the bottom scores for all but one indicator. Moreover, the scores of Heroku are more than 20 times as high for each indicator. The results of the remaining three PaaS providers; CloudFoundry, Engine yard and OpenShift, contain some intriguing patterns relative to one another. CloudFoundry strictly has first and close second positions. Engine Yard is a close second for half of the indicators and the overwhelming third for the other half. OpenShift has a mix of all three places. However, its low scores for f (number of followers) and h (repositories updated at least once) are higher than both CloudFoundry and Engine yard relative to the number of repositories these metrics are derived from. This is further confirmed by OpenShift having the highest score for indicator j, updated vs. non-updated ratio.

Based on metric a, Figure 1 was created. For each PaaS, it displays the relative number of active developers in a weekly period compared to the total number of active developers in the past year. The general trend of the industry is positive. Developers of most PaaS providers have created or updated more repositories in recent weeks than in weeks at the start of the year. OpenShift in particular has a steep graph upwards throughout the year, with just one third (34.29%) of developers in the first half of 2012. However, Google App Engine and Engine Yard display a negative pattern. More than a quarter (26.37%) of Google App Engine developers have not updated their repository since the first quarter of 2012. Engine Yard is in a similar position, with the majority of their developers (52.04%) in the first six months of 2012.

7 Discussion of PaaS Ecosystem Health

The analysis of all indicators introduced Heroku as the dominant, leading PaaS. The scores of the runner-up PaaS, Google App Engine, further confirms this dominance. Only two sub-indicators exceed half the scores of Heroku.

The passive interest shown by the number of followers is four times greater for Heroku than the passive interest of Google App Engine and an order of magnitude greater than all the others. This indicates that interest in Heroku is not limited to active developers, but includes a large number of passive GitHub users that are interested in the progress of these projects. Based on these observations, we expect Heroku to maintain its dominance in the coming years.

However, absolute size is not an indicator of a great ecosystem per se. OpenShift its good performance on indicators a, i and j illustrates this. OpenShift has results close to Azure for many indicators, even though Azure boasts a larger customer-base and a three times as large community. Moreover, the active developers for OpenShift in the past shows the most promising slope of all providers, indicating a good position to grow in the PaaS industry in the future. An explanation for this growth in success is that Openshift is a subsidiary of Red Hat, which has a large existing customer base and a devoted community.

The general positive distribution of active developers per week implies positive future growth expectations for the PaaS industry as a whole. However, the shift in the graphs of Google App Engine and Engine Yard indicates that developers may already be abandoning these PaaS providers, resulting in a future with a compounding loss of developers and ultimately end of business.

Do these results commend or discourage doing business with certain PaaS providers? Assumptions need to keep the origin of the data in mind. Data collected from GitHub might not be representative for commercial private cloud providers, e.g. Azure, as those developers are more inclined to use proprietary solutions or private repositories to collaborate. Furthermore, should PaaS providers with negative indicators be neglected? Although e.g. Engine Yard scored low, they reported revenues of 28$ million in 2011. This shows an obvious market interest and a potential for healthy future growth. However, business aspects as revenues or customer size are not publicly available or verifiable for the majority of PaaS providers. As a consequence, we are unable to evaluate the *long-term financial well-being of the business ecosystem* and are restricted to the *long-term strength* aspect of the ecosystem health definition on page 184.

8 Conclusion and Future Work

This exploratory research provides businesses with a method to evaluate PaaS providers. The current ecosystem health is skewed to two major players, with Heroku far ahead. If OpenShift can maintain its growing trend, this will change in the future. The PaaS industry is still young, new businesses will enter the market and others will exit. Additionally, PaaS providers can evaluate the current state of their ecosystem and adjust their corporate strategy accordingly.

Future research could validate our method by confirming whether weak PaaS providers went out of business or by applying this model to other players. To assist academics and businesses in this process, we provide the data extraction methods and analysis code on GitHub. Furthermore, retrospective studies with a broad scope can document the developments within the PaaS industry as a whole.

References

1. Gartner Says Worldwide IT Spending On Pace to Surpass 3.6 Trillion in (2012), http://www.gartner.com/it/page.jsp?id=2074815
2. Mell, P., Grance, T.: The NIST Definition of Cloud Computing (draft). NIST Special Publication 800-145 (2011)
3. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud Computing, The Business Perspective. Decision Support Systems 51, 176–189 (2001)
4. 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)
5. van de Zande, T., Jansen, S.: Business Continuity Solutions for SaaS Customers. In: Regnell, B., van de Weerd, I., De Troyer, O. (eds.) ICSOB 2011. LNBIP, vol. 80, pp. 17–31. Springer, Heidelberg (2011)
6. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Business Review 82(3), 68–81 (2004)
7. Kittlaus, H.B., Clough, P.N.: Software Product Management and Pricing: Key Success Factors for Software Organizations. Springer, New York (2009)
8. Bosch, J.: From Software Product Lines to Software Ecosystems. In: Proceedings of the 13th International Software Product Line Conference, pp. 111–119. ACM, New York (2009)
9. Jansen, S., Finkelstein, A., Brinkkemper, S.: A Sense of Community: A Research Agenda for Software Ecosystems. In: 31st International Conference on Software Engineering, New and Emerging Research Track, pp. 187–190. IEEE Press, New York (2009)
10. Spriensma, G.J.: 2012 Year in Review, http://www.distimo.com/publications/archive/Distimo%20Publication%20-%20Full%20Year%202012.pdf
11. Crowston, K., Howison, J., Annabi, H.: Information systems success in free and open source software development: Theory and measures. Software Process: Improvement and Practice 11(2), 123–148 (2006)
12. 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)
13. Stewart, K.J., Gosain, S.: The impact of ideology on effectiveness in open source software development teams. MIS Quarterly 30(2), 291–314 (2006)
14. Krishnamurthy, S.: Cave or community? An empirical examination of 100 mature open source projects. First Monday 7(6) (2002), http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1477/1392
15. Crowston, K., Annabi, H., Howison, J., Masango, C.: Effective work practices for software engineering: free/libre open source software development. In: Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research, pp. 18–26. ACM Press, New York (2004)

Defining App Stores: The Role of Curated Marketplaces in Software Ecosystems

Slinger Jansen and Ewoud Bloemendal

Utrecht University
slinger@slingerjansen.nl,webloem@gmail.com

Abstract. The app store is a novel concept in the software business, that has changed the way in which customers perceive software and its day-to-day use. The concept, however, is poorly understood, which can be observed by lack of a comprehensive definition and relatively little literature on the topic. This paper provides a definition of app stores, provides a conceptual model of the concept, and supplies typical features and policies that are observed in app stores, using six case studies. The increased understanding that the research provides, aims to help practitioners make their app store more successful and provides researchers with a frame for defining and analyzing app stores.

Keywords: App stores, software ecosystems, comparative multiple case study.

1 Introduction

The product software business is a fast changing business. One of the most powerful changes that the software business currently is experiencing is the introduction of app stores, which are marketplaces for applications that are available for instant download. App stores are influencing the industry in the following major ways. First, people are becoming more aware of the software business: with so much software available in everyone's pocket, increasing numbers of people are exposed to the app business. Furthermore, due to the low prices of apps in the app store, business models are radically changing to constantly add value to the product such as content, as to generate equal amounts of revenue from complex software systems as was possible in the "old days" of license and maintenance models. Thirdly, app stores appear to be the method of choice to build up a healthy software ecosystem [10,8,6]. It is surprising that this topic has not received more attention over the past years. In this paper we aim to fill that gap, by providing an exhaustive overview of the features and policies that determine the structure of an app store. Furthermore, we provide the following definition of app stores.

> **App store:** *An online curated marketplace that allows developers to sell and distribute their products to actors within one or more multi-sided software platform ecosystems.*

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 195–206, 2013.

Jansen, Finkelstein, and Brinkkemper define a software ecosystem as a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts [9]. An app store can be seen as a catalyst in such a software ecosystem. The services it offers are part of the common platform of the ecosystem and it can have a pivotal role in creating the common market. App stores allow developers to monetize their software and bring consumers new functionality. A successful app store is beneficial to the success of a software ecosystem which consequently can be beneficial to the company that owns it, or according to a Deutsche Bank analyst, Apple's app store is the "the competitive moat which competing handset vendors cannot replicate" [3].

The literature on app stores is limited, possibly due to the relative short existence of the phenomenon. The most well-known app stores come from the domain of mobile phone platforms, thus it is no surprise that these app stores have received the most attention in literature. The success of Apple's app store has led to a fair amount of literature [2,10] specifically about the iOS ecosystem. A broader perspective is given by Holzer and Ondrus [5] when they take a developer view on the significant structural changes in the mobile application market. They introduce eight propositions on how the mobile software market changes for developers. Their first proposition is that portal centralization (their way of describing the introduction of an app store) makes access to customers easier. They also propose that portal centralization lowers distribution costs but also limits the freedom of developers. With regard to mobile ecosystems based on open technologies, they propose that open technology will lower the development costs of applications and offer more job opportunities for developers. When app stores choose to support a larger number of devices the authors propose that this higher device variety increases freedom for developers but also increases customization cost. Finally Holzer and Ondrus [5] propose that a fully integrated platform facilitates the flow through the distribution process.

At the moment no literature on app stores has been found that combines a broad software ecosystem scope with the perspective of an (aspiring) app store owner and therefore this study will try to fit in this gap. The objective for this study is help app store owners gain more insight in the app stores by creating an app store definition and performing a multiple case study on app store features and policies.

In order to provide more insight into app stores and their features and policies, this paper uses the following structure. In Section 2 an app store definition is introduced together with a conceptual model of an app store. In Section 3 the case study method is explained, the cases are introduced, and we provide a brief discussion on validity of the research. In Section 4 the models resulting from the case studies are listed with a table of feature and policy descriptions. The study concludes with the key findings, limitations and opportunities for further research. The highlights of the research are the app store definition, the policies

and features, and the finding that most app store coordinators are aiming to cover all features and policies instead of consciously deciding to leave out features and policies.

2 Conceptual Model: Defining App Stores

We introduce the following definition of an app store: *An online curated marketplace that allows developers to sell and distribute their products to actors within one or more multi-sided software platform ecosystems.*

The first element of the definition is *online curated marketplace.* A marketplace is a common word describing a location where goods and services are exchanged. The adjective curated introduces the concept of a curating party that organizes and selects the collection of apps in an app store, a task generally performed by the owner. The word of this element was added to distinguish the app store from a brick and mortar stores. *Software ecosystems* are part of the definition to emphasize the relation between an app store and its ecosystem. It the following part two groups of users are identified: *software developers* and *users of a software platform.* The existence of these two groups are typical for an app store and thus part of the definition. Using this definition, a list of requirements is made to limit the scope of what is considered an app store in this research. To be considered an app store a system should: (1) be available using the internet, (2) be curated by an organization, typically but not necessarily the platform owner, (3) allow for the selling and buying of software products, (4) take care of the financial transactions involved in selling the software products, (5) have two distinct user groups: developers and users, (6) be serving one or more software ecosystem, and (7) implement a platform that takes care of the distribution of the software products. Please note that an app store can apply to one ecosystem, such as Google Play serving the Android ecosystem, or multiple ecosystems, such as the BinPress app store, where code can be purchased for multiple platforms and ecosystems.

App stores would not exist if it were not for platforms: a set of solutions to problems that is made available to the members of the ecosystem through a set of access point or interfaces [7]. Furthermore, Hagiu and Wright speak of a multi-sided platform as being an organization that creates value primarily by enabling direct interactions between two (or more) distinct types of affiliated customers [4]. Three platform types are distinguished by Baldwin and Woodard [1]: platforms in a firm as product lines, platforms across multiple firms as multi-product systems, and platforms in the form of multi-sided markets. They argue that a platform has a platform architecture and corresponding design rules that governs the relations between components of the platform and allow them to interoperate. According to the authors this architecture shows a fundamental unity for each type of platform. They describe this unity in platform architectures as "modularizations of complex systems in which certain components (the platform itself) remain stable, while others (the complements) are encouraged to vary in cross section or over time.". According to Baldwin and Woodard, the

most stable element of a platform are the interfaces between the platform and its complements, even more stable than the core of the platform itself.

In order to create a better understanding of how the app store acts within its software ecosystem a conceptual model of app store mechanics is proposed in Figure 1. In the ovals the different actors within the ecosystem are modeled. The first one on the bottom is the owner, which is not necessarily the owner of the ecosystem but rather the owner of the app store. The other two ellipses both represent a set of actors rather than one actor. The first set of actors are the end users. The second set of actors are the developers. The triangular shape represents the app store, with each edge facing one of the actors in the ecosystem. The app store functions as a marketplace bringing users and developers together. Developers can publish their apps using the app store, end users can search for apps and buy them from the developers using the app store. The app store is created and governed by the owner who generally takes a share of the generated revenues as compensation for this work. These relations are represented by the arrows from and to the app store triangle.

Within the triangle three concepts are depicted: two bottom squares *features* and *policies* and at the top the *characteristics* square. Features represent individual parts of the software systems that the actors can interact with. Policies represent the rules, regulations and governing processes that limit the functional reach of the features. Features and policies together form the part of an app store that the owner can directly influence. The characteristics square represents a set of app store characteristics that cannot be directly influenced by the app store owner. An example of a such characteristic would be the total number of apps available in an app store. Other examples of these characteristics are the number of developers, the number of end users, the quality of the apps or the usability of the app store software. The arrows going from the features and policies to the characteristics represent the assumption that app store owners try to influence the characteristics by implementing a certain set of features and policies.

3 Research Approach: Identifying Features and Policies

In order to identify features and policies in existing app stores, case studies were performed using a multiple case study method based on the case study methods of Yin [11]. At first, a case study protocol was created to ensure a consistent research execution amongst the cases. Secondly, a long-list of existing app stores was created based on a set of web search queries, resulting in a list of 81 app stores. The following criteria were used to create a convenient sample for the research. The first criterion checked whether the researcher had access to the app store and could get a complete overview of the features and policies. The second set of criteria aims to improve the generalization of the sample by selecting app stores from multiple vendors and only allowing app stores with a minimum of 1000 transactions and existing longer than 6 months. This resulted in the following six app stores that were selected for the case study: Google Play, SlideMe, Apple Appstore, Binpress, Amazon app store for Android and Intel AppUp.

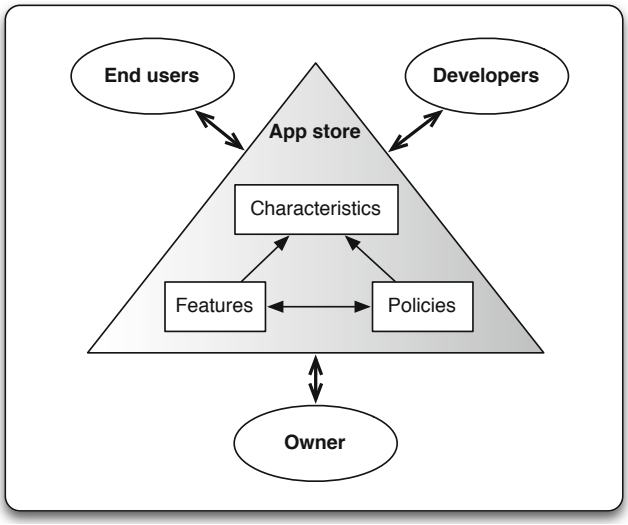

Fig. 1. Conceptual Model of an App Store

Table 1. Growth of the Data Set in Different Phases of the Research

Research Step	# Features	# Policies	# Apps	Paid (%)
Initial model (from literature)	14	8		
After Case 1: Google Play	28	10	450,000	28%
After Case 2: SlideMe	35	17	20,000	26%
After Case 3: Apple Appstore	40	21	650,000	66%
After Case 4: Binpress	57	20	30,000	65%
After Case 5: Amazon appstore	64	24	369	100%
After Case 6: Intel AppUp	67	24	3,000	64%

The goal of the six case studies was to create an overview of the features and policies of existing app stores that accurately describes the features and policies of an app store. For each of the six selected app stores data was gathered on its features and policies. Data for the case studies was gathered by gaining access to the app store and making screenshots to of features. Documentation was used and collected from three different sources: by using the support documentation provided by the app store owner, by directly searching tech blogs and news websites, and lastly by blogs referred to by general web searches.

For each case a case study database was created in which all documents were stored. The data in the database was then analyzed and the observed features and policies were filled in and if needed features or policies were added, removed or renamed. Whenever possible data from multiple convergent sources was used to determine the existence of a feature or policy. As Yin suggested the case studies were done in a sequential order where each case study used the results

of the previous case study as a starting point. This way each case study resulted in a more refined and more extended version of the model. In order to create a starting point for the first case study a preliminary model was created using blog posts comparing app stores. In Table 1 the growth of the model through this process is shown. After the completion of the model all previous cases were revisited in order to complete the dataset.

3.1 Validity

First of all **construct validity** of the research was improved by, whenever available, using multiple sources of evidence in a convergent manner. In practice this meant that the existence of a feature or policy was underpinned by both direct observations (screenshots) and documentation provided by the app store owner or a third party. Also a chain of evidence was established for each feature and policy in each case study. Using references to the document database each observation can be followed back to its underpinning evidence.

The **external validity** of the app store classification model was improved by the use of replication logic. The created model is based on six consecutive case studies and was able to describe all observed features and policies. Also the statistics in Table 1 show that the model did not need many changes between the fifth and the sixth case suggesting that the model is approaching completeness. Of course performing more case studies until the model would not be refined anymore would always improve the external validity of the classification model.

The **reliability** of the case study results was improved by creating a case study protocol. This case study protocol ensured that the used field procedures stayed the same between the different cases. The reliability was further improved by creating a document database for each case study. When data was gathered it was consistently added to the document database before any further analysis. The document database allows the analysis of the research to be repeated based on the same data. This proved valuable when the cases had to be revisited with the completed model. In many cases no additional data had to be gathered even though the model had significantly grown.

3.2 Case Descriptions

The first case study was done on Google Play, launched in 2008 and now the biggest app store in the Android ecosystem. It serves the Android ecosystem, an open source operating system for mobile devices and tablet computers. From its inception in 2008 it allows developers to sell applications and games to end users. After its merger with Google Music it was re-branded from Android Market to Google Play and it became a digital multimedia content service that also sells books, music and movies.

The second case is SlideMe app store, which launched in April 2008 and introduces itself as the third app store in the Android ecosystem, after Google Play and the Amazon Appstore. SlideMe provides an alternative app store for devices that for some reasons cannot install Google Play and is used by over 120

OEM devices as their alternative app store. SlideMe does not charge transaction fees to developers other than the payment processing fees because SlideMe earns its money by providing white-label curated app stores for OEM devices, i.e., manufacturers can use the SlideMe software and rebrand it as their own for a service fee.

The third case is the Apple app store, launched in July 2008. The Apple app store is the most successful app store in the market of mobile devices and is probably the best known too. The app store hosts applications for mobile devices made by Apple running the iOS platform, namely the different versions of the iPhone and iPad tablet. From the start the Apple Appstore was integrated with their already popular iTunes music download service. Under app developers the Apple appstore is notorious for its strict approval process that apps have to go through before publishing. It was also the first app store featuring in app billing and content subscriptions.

The fourth researched app store is Binpress, founded in 2011 as a marketplace for source code. Binpress hosts source code for a variety of twelve programming languages with 16 frameworks and 18 platforms. Because Binpress is a marketplace that sells and buys from developers to developers it offers features tailored to developer needs, such as issue tracking and component support forums.

The fifth app store used as case is the Amazon app store for Android which opened in March 2011. It is the second most important app store in the Android software ecosystem and was founded by online retail giant Amazon. The app stores website piggy backs on the existing web retail infrastructure of Amazon and its massive user and credit card details database. The most eye catching features that distinguish the Amazon app store for Android from Google Play are the availability of the Deal of the Day promotion and the possibility to test drive apps before buying.

The last case study is the app store introduced in January 2010 by semiconductor manufacturer Intel called Intel AppUp. The initial focus of the Intel AppUp store was on applications for the at that time popular netbook devices using the Intel Atom processor platform. Nowadays the Intel AppUp store hosts apps for platforms Adobe Air, Microsoft Silverlight, the Linux-based MeeGo operating system for mobile devices and native Windows XP and Windows 7 applications.

4 Case Results

After all features and policies were identified and the data for each of the case studies was available a set of common features shared by all app stores could be created. This corresponds with the fact that all app stores were selected using the same definition and set of criteria. The fifteen core features are shown in Table 2. Each core feature can be mapped to a part of the app store definition: ‘app categories’, ‘app listing’, ‘app lists’, ‘featured apps’, ‘ratings’, ‘reviews’ and the ‘search’ feature can be mapped to the ‘online curated marketplace’ part, ‘developer app management’, ‘developer transaction list’, ‘distribution integration’,

Table 2. Core app store features

Core feature	Descriptions
app categories	Apps are listed in categories and subcategories
app listing	Apps are listed with full description, images, etc.
app lists	apps are listed, e.g. top selling lists or latest additions
dev app management	Devs can manage their apps in a developer console
dev transaction list	Devs can manage their transactions
distribution integration	Distribution and installation happens through platform
featured apps	Apps can be featured to receive more attention
free revenue model	Apps can be offered for free
paid revenue model	Apps can be sold
pay out methods	Number of pay out methods
payment methods	Number of payment methods
platform comp. filter	Apps have information on their platform compatibility
ratings	Apps can be rated by the user
reviews	Users can read and write reviews of an app
search	Users can search for apps using search keywords

'free revenue model', 'paid revenue model', 'pay out methods' and 'payment methods' map back to 'allows software developers to sell and distribute their products to users' and 'platform compatibility filter' maps back to the 'software platform' part. In Table 2 the descriptions for each of the core features are listed.

To give the elements of the model a meaningful categorization, the app store characteristic that is mainly influenced by each feature or policy was determined. This resulted in a categorization based on the following nine app store characteristics: app store usability to the user, app findability, app quality, developer quality, app store usability to the developer, app visibility, monetization potential, user interaction, openness. These characteristics could further be divided into user focused characteristics and developer focused characteristics. The accompanying feature and policy descriptions can be found in Tables 3 and 4. The models and the descriptions give a complete overview of the observed features and policies in the six case studies.

5 Analysis of the Results

In Figure 2 a sample of the table is found that lists the features and policies of the app stores under study. The main finding from these data, is that most of the app stores are on their way to include most, if not all features and policies in their app stores. Furthermore, due to the sheer size of the effort of implementing an app store we expect to see third parties offering white-label app store platforms[1], thereby reducing effort for platform owners in orchestrating the ecosystem.

Some other observations can be made, however, about the data. To begin with, the core features are supported by all app stores, which is one of the reasons why

[1] Such as the start-up `www.appgalleries.com`

Table 3. User and developer centric app store features

User focused: app findability	
recommendations	Apps are recommended based on user profile
store curation tags	Developers can tag and categorize their apps
User focused: app quality	
app security integration	An app platform security system is provided
app security reporting	Harmful apps and security threats can be reported
app test driving	Apps can be test driven before purchase
content rating filter	Apps are rated with a content rating
device compatibility	Apps can be filtered on device compatibility
remote app remove	Harmful apps can be removed by owner from device
user review curation	Users can curate each other's reviews
User focused: app store usability	
automated refunds	Users can apply for refunds
developer refunds	Developer can initiate refunds
device integration	Devices have the app store installed by default
multi language	app store is internationalized
multichannel dist.	Users can use multiple channels to acquire apps
multi-currency	Multiple currencies are supported
update integration	Automated updates are possible for the app
user app list	A list of apps downloaded or purchased by a user is available
user subscription list	A list of all content subscriptions of user is available
user transaction list	A list of all transactions made by a user is available
Developer focused: feedback potential	
app suggestions	Users can leave suggestions for new apps
app support forums	Each app has its own support forum
beta testing mgmt	Developers can invite beta testers for their apps
feature suggestion	Users can suggest features to apps
issue tracking	Users can report issues and track their progress
user profile	Users have extended user profiles
Developer focused: monetization potential	
affiliate program	Users can make money directing "traffic" to apps
affiliate stores	Apps are offered through other channels
component offering	Developers can also offer separate components
discounts	Apps can be temporarily offered at a discount
in-app advertising	Monetization can also be done through advertising
in-app billing	Users can purchase extra features and content in-app
licensing integration	License checking for illegally installed or acquired apps
social media sharing	Apps can be shared through social media
subscriptions	Developers can offer content subscriptions to users
volume pricing	Developers can offer volume pricing
Developer focused: app store usability	
data API	Developers can get data from the app store using an API
deployment integration	Developers can automate the deployment to the app store
dev contract mgmt	Developers can manage contracts with the app store owner
dev multi-user login	Multiple users can be added to a developer account
dev sales statistics	Developers have access to sales statistics
geographic targeting	Apps can be targeted to geographic regions
tax support	The app store applies legally required taxation
Developer focused: visibility	
cross selling	Associated apps are shown to the developer
developer app list	A list of apps made by each developer is available
developer profile	Developers have profile pages with details

Table 4. User and developer centric app store policies

User focused: app quality	
approval before publish	Apps are checked by the store owner for compliance manually
automated monitoring	The app store uses an automated system to check for apps that do not comply
code quality curation	The quality of the code of apps is checked
functional quality curation	The functional quality of apps is curated by the owne
interface quality curation	The owner checks apps for compliance with interface guidelines
review after purchase	Reviews for an app can only be posted by users that have downloaded or purchased the app
review poster verified	Users that are verified by the app store owner can post reviews
User focused: developer quality	
developer verification	Developers have to prove their identity to the app store owner before receiving payments
recurring fee	A recurring fee is required to be a developer at an app store
Developer focused: monetization potential	
pay-out delay	The delay between the payout and the last day of the scheduled date range
pay-out schedule	The schedule payment schedule of the revenue share of the sales to the developer
pay-out threshold	The minimum amount required to be eligible for a payout
price control	The party that can control the price of an app
revenue share	The percentage revenue share that goes to the developer
third party app stores	Apps are allowed to reference other app stores
third party in-app advertising	Apps are allowed to use third party in-app advertising
app store refunds	The app store owner has a clear refund policy and provides refunds on request of a user
third party in-app billing	Apps are allowed to use a third party system for in app purchases
Developer focused: openness	
competing functionality curation	Apps that have features that compete with the app store owner are not approved
custom licensing	Developers can provide their own custom EULA, not limited by the app store owner
guided licensing	The app store owner provides and enforces guidelines for EULAs
open source licensing	Developers can use open source licenses to publish their apps
Developer focused: visibility	
geographical availability	The number of countries an app store is available in

	Intel AppUp	Binpress	Amazon appst	Apple appstor	SlideMe	Google play
app security integration	no	na	yes	na	yes	yes
app security reporting	yes	na	yes	no	yes	yes
app test driving	no	no	yes	no	no	no
content rating filter	yes	no	yes	yes	yes	yes
device compatibility filter	partial	na	partial	partial	partial	yes
remote application removal	no	na	no	yes	no	yes
user review curation	no	yes	yes	yes	no	yes

Fig. 2. Sample of the feature and policy evaluation for each app store case

they made the short list. The data differs greatly, however, for all other features and policies. The largest outlier is Binpress (the source code app store), since it supports many different features and policies than the other app stores.

At the time of writing only one app store (Amazon) out of six enables end-users to test drive an app before purchasing. Secondly, besides Binpress, all app stores include a content rating filter, since Binpress is the only source code app store, in comparison to the others. Only Google Play at this point supports automated developer and end-user refunds, suggesting that the Google Play billing system may be the most advanced, which we do not find surprising when considering Google's e-payment strategy and product portfolio.

In regards to feedback management, very little features are supported. Only one app store supports app suggestions, and only one other supports beta testing management. Yet another app store supports issue tracking and feature suggestions, which shows that developers could be supported much more extensively by the app store owners than they are now. In regards to monetization there is a varied set of app stores supporting the monetization features, and no generalizations can be made about those features at this time. It is interesting, however, to see that only one app store enables component (i.e., supporting app development) sharing.

When looking at the specific app stores it can be observed that Google Play and Amazon's app store are the most complete in supporting developers and end-users. SlideMe supplies the least features and could be considered the most immature. Surprising is that both in terms of developer features and end-user features the Apple app store is not the most mature, even though it is the 'benchmark' app store that significantly increased popularity of app stores.

6 Conclusions and Further Research

The objective for this study was to help app store owners gain more insight in the app stores by creating an app store definition and performing a multiple case study on app store features and policies. To achieve this objective, first, the definition of an app store was given. Section 4 shows that the common features observed in the case studies could be mapped to the different parts of this definition. Secondly, in section 2, the conceptual model features and policies were defined: features represent individual parts of the app stores software systems that the actors can interact with. Policies represent the rules, regulations and

governing processes that limit the functional reach of the features. The description tables of these features and policies can be used to identify a feature or policy in an app store.

The app store definition combined with the feature and policy models provide app store owners or organizations considering becoming an app store owner with insight into the concept. One possible problem with the external validity might be the number of features and policies that were not applicable to the source code app store Binpress. In order to improve the external validity of the model for source code app stores it would be useful to add more case studies on source code marketplaces, or even classify app stores along their content, features, and policies. These explorations into domain specific app stores are seen as future work.

The possible relation between the strategy of app store owners and the app store characteristics is a topic for further research. Possibly, one could formulate sets of policies and features based on typical strategies followed by platform owners. However, for such an exploration, more insight into platform owner strategy and success is required. One challenge in such analysis is that a platform strategy, of which an app store is a small part, may be much more influential to the success of the platform than the app store.

References

1. Baldwin, C., Woodard, J.: The architecture of platforms: A unified view. In: Gawer, A. (ed.) Platforms, Markets and Innovation. Edward Elgar Pub. (2010)
2. Eaton, B., Elaluf-Calderwood, S., Sørensen, C., Yoo, Y.: Dynamic structures of control and generativity in digital ecosystem service innovation: the cases of the apple and google mobile app stores. LSE, London Report 44(183), 1–25 (2011)
3. Elmer-DeWitt, P.: 6 ways iphone and android users differ. Fortune. February 25 (2010)
4. Hagiu, A., Wright, J.: Do you really want to be an ebay? Harvard Business Review (2013)
5. Holzer, A., Ondrus, J.: Mobile application market: A developers perspective. Telematics and Informatics 28(1), 22–31 (2011)
6. Hyrynsalmi, S., Mäkilä, T., Järvi, A., Suominen, A., Seppänen, M., Knuutila, T.: App store, marketplace, play! an analysis of multi-homing in mobile software ecosystems. In: Proceedings of the International Workshop on Software Ecosystems, p. 59 (2012)
7. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Bus. Rev. 82(3) (2004)
8. Idu, A., van de Zande, T., Jansen, S.: Multi-homing in the apple ecosystem: why and how developers target multiple apple app stores. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, pp. 122–128. ACM (2011)
9. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: 31st International Conference on Software Engineering-Companion Volume, ICSE-Companion 2009, pp. 187–190. IEEE (2009)
10. West, J., Mace, M.: Browsing as the killer app: Explaining the rapid success of apple's iphone. Telecommunications Policy 34(5), 270–286 (2010)
11. Yin, R.K.: Case Study Research: Design and Methods, 3rd edn. Applied Social Research Methods Series, vol. 5. Sage Publications, Inc. (December 2002)

Towards Platform-Based Enterprise Systems – Conceptualization and Research Directions

Carl Simon Heckmann[1] and Alexander Maedche[1,2]

[1] Institute for Enterprise Systems, University of Mannheim,
[2] Chair of Information Systems IV, Business School, University of Mannheim,
68131 Mannheim, Germany
{heckmann,maedche}@es.uni-mannheim.de

Abstract. Historically different approaches such as tailor-based functional systems, Enterprise Resource Planning or SOA-based systems have been implemented to provide IT-based support for business processes. They all share the tension between standardization and differentiation. Similar trade-offs have been discussed in product development literature proposing modularization and platform-based approaches as a potential solution. Applying these concepts to the context of enterprise systems (ES), this work provides a first conceptualization of platform-based ES (PBES) and suggests possible directions for future research.

Keywords: Enterprise System, Platform, Modularization, Product Development.

1 Introduction

The debate whether and to what degree IT can be the source of competitive advantage is still ongoing [1, 2]. Some parts of corporate IT can be viewed as a commodity [1], while other parts, in particular these supporting differentiating business processes, should be viewed as an enabler for achieving competitive advantage [3]. Enterprise Systems (ES) represent a specific category of information systems. They build on pre-packaged industry best practices embedded in standardized product software and target large-scale integration of data and business processes across all company's functional areas and beyond company borderlines. . Specifically the ERP concept came with the promise of tight integration [6] and standardized best practices for a wide range of processes [7]. However, this standardization leads to high costs and effort for customization required for differentiation [8]. Service-oriented architecture (SOA) were introduced to allow ES to be adapted more flexibly, and to better meet companies business requirements, allowing to differentiate from competitors. However in practice several shortcomings of the SOA paradigm have been identified, e.g. increased system complexity [3]. Thus, companies need to decide on the degree of standardization and differentiation within their ES approach. This continuum reaches from highly standardized systems to differentiating best-of-breed approaches. The latter one allows less expensive differentiating, however overall efficiency is reduced due to the lack of standardization.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 207–211, 2013.

In other engineering disciplines similar trade-offs have been addressed using modularization and platform-driven product development [9]. Adapting these concepts to the context of ES provides new insights and is consistent with previous works, which have characterized ES as modular systems [10, 11]. Furthermore, consumer-oriented IT platforms, such as iOS, already show the appropriateness of platform concepts to the IS domain [12]. The concept of platform-driven product development of ES is largely unexplored so far leading to the following research question: *What are the characteristics of a platform-based enterprise system (PBES)?*

By addressing this research question we intend to contribute to the understanding of platform-concepts in the ES context by providing a first conceptualization of PBES and to highlight possible further research directions.

2 State-of-the-Art

This body of work draws on the extensive literature streams in the area of ES as well as modularization theory and platform-driven product development. This section provides an overview to foster a comprehensive understanding of challenges in ES and how these can be addressed by applying the platform paradigm.

2.1 Enterprise Systems

ES emerged as response to high costs and limited integration of custom-developed systems. They are characterized as large-scale organizational systems built around packaged software embedding best-practices and a high degree of configurability and customizability [7, 8]. The various benefits include operational improvements through process automation and best-practices as well as enhanced decision-making [7, 8]. However, best practices may not fit the actual practices of a particular company requiring expensive customization of the system. Thus a "best-of-breed" strategy is followed in many companies to minimize customization by selecting systems which better fit their business requirements [8, 13].

There is some dispute whether IT can be the source of competitive advantage [1, 2]. Recent studies indicate that there are parts of IT which are enabler for differentiation potentially leading to competitive advantage [3]. In today's highly uncertain and dynamic environment ES need to be adapted flexibly to better meet changing requirements while achieving efficiency through standardization [2]. Integration and modularity are mentioned as requirements for flexibility, however it is not stated how this can be achieved [3].

Previous and current generations of ES failed in solving the tension between standardization and differentiation. As comparable trade-offs have been discussed extensively in product development literature concepts such as platform-driven product development and modularization are transferred to the context of ES to provide insights on how to overcome the challenge of creating integrated, yet flexible ES.

2.2 Platform-Based Product Development

In product development, modularization provides means to manage complexity by breaking up a system into discrete chunks that communicate through standardized interfaces [12, 14, 15]. This leads to increased reusability and flexibility through a larger variety of configurations, resulting in reduced product and switching cost [15]. Modularization also allows for rapid and steep performance gains through recombination [14], decreased innovation costs and improved innovation outcomes [14].

Products in engineering disciplines are complex systems, which are defined as "one made up of a large number of parts that interact in a non simple way“[16]. Complex systems are composed of interacting modules that are always to some degree interdependent and inhibit high synergistic specifity, posing a strong force against modularization [15]. The set of modules that is used or reused across implementations comprises the platform [17]. In the IT context a 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 (e.g., Apple’s iOS)” [12].

This state-of-the-art section presented the challenge to balance standardization and differentiation in previous and current generations of ES. Similar trade-offs have been addressed in other engineering disciplines by the use of platform-based product development concepts. As platform concepts such as iOS have already been successfully transferred to the IT domain in consumer settings, we will explore how such concepts can be applied to ES.

3 Conceptualization and Research Directions

Current ES do not solve the tension between standardization and differentiation. Product development in other engineering disciplines such as the automotive industry already addressed this tension [9]. However there are differences between ES and traditional physical products. Compared to physical products ES are of dynamic nature as they are adapted to changing business requirements and evolve throughout their life cycle [7]. ES are complex systems involving many different elements addressing a wide area of business requirements in a company. Such complex systems are characterized as near-decomposable systems [16]. These elements show the characteristic of high synergistic specifity posing a force against modularization [15]. Changing one of these modules leads to changes in a number of other modules. We propose that these modules, inhibiting high synergistic specificity, should be standardized as change is too costly. Following the previously introduced platform definition [17], we characterize the set of modules, which cannot be separated from each other, as the core platform of the ES.

On the basis of the definition of ES provided by Davenport [6] and the presented related work we propose the following definition: “A platform-based enterprise system (PBES) consists of

- a standardized core containing modules fulfilling high standardization requirements and
- a flexible sphere containing complementary modules

to enable the company to realize standardization benefits for their stable core systems while still allowing to differentiate. ”

This first understanding of PBES allows identifying potential directions for further research. Following the product development literature [9] a rationale for allocating modules to the standardized core platform flexible sphere is required. Consequently the following research question needs to be answered:

What are the determinants for the allocation of modules either to the core platform or the sphere of complementary modules?

This automatically leads to another interesting consideration with regards to modules. What is the ideal size of a module [18]? Too coarse-grained modules prevent clear allocation [14], while too fine-grained modules result in high coordination effort, leading to the following research question:

What is the effect of the application system modularity on process performance and how is this effect influenced by different integration requirements of modules?

Knowing which modules are part of the platform and which modules are part of the complementary set of modules as well as their granularity leads to the next question which is of rather technological nature. Current developments such as PaaS can provide valuable insights in how PBES can be implemented [12].

What are the requirements of a platform to allow complementary modules to be added in a flexibly way to the core platform?

4 Conclusion

Different generations of ES tried to solve the tension between integration and flexibility but have failed so far. As similar trade-offs have been identified in product development literature this body of work transferred the platform approach to product development to an ES context. This paper provides two contributions. One the one hand we highlight an important area in ES design, namely how integration and flexibility can be addressed in parallel by applying the platform approach to product development to the ES context. On the other hand this research contributes a first conceptualization of PBES as a starting point for future research.

References

1. Carr, N.G.: IT Doesn't Matter. Harvard Business Review 81(5), 41–49 (2003)
2. Ward, J.M.: Information systems strategy: Quo vadis? 20th Anniversary Special Issue 21(2), 165–171 (2012), doi:10.1016/j.jsis.2012.05.002
3. Gebauer, J., Lee, F.: Enterprise System Flexibility and Implementation Strategies: Aligning Theory with Evidence from a Case Study. Information Systems Management 25(1), 71–82 (2008), doi:10.1080/10580530701777198
4. Maedche, A., Mueller, B.: Enterprise Systems-A Research Agenda. SSRN 2033145 (2012)
5. Orlikowski, W.J.: The sociomateriality of organisational life: considering technology in management research. Cambridge Journal of Economics 34(1), 125–141 (2010), doi:10.1093/cje/bep058
6. Davenport, T.: Putting the enterprise into the enterprise system. Harvard Business Review 76(4) (1998)
7. Lynne Markus, M.: Cronelis Tanis The Enterprise System Experience - From Adoption to Success. In: Framing the Domains of IT Management Projecting the Future Through the Past, pp. 173–207.
8. Shang, S., Seddon, P.B.: Assessing and managing the benefits of enterprise systems: the business manager's perspective. Information Systems Journal 12(4), 271–299 (2002), doi:10.1046/j.1365-2575.2002.00132.x
9. Halman, J.I., Hofer, A.P., van Vuuren, W.: Platform-Driven Development of Product Families: Linking Theory with Practice. Journal of Product Innovation Management 20(2), 149–162 (2003)
10. Davenport, T.H., Harris, J.G., Cantrell, S.: Enterprise systems and ongoing process change. Business Process Management Journal 10(1), 16–26 (2004), doi:10.1108/14637150410518301
11. Fan, M., Stallaert, J., Whinston, A.B.: The adoption and design methodologies of component-based enterprise systems. European Journal of Information Systems 9(1), 25–35 (2000)
12. Tiwana, A., Konsynski, B., Bush, A.A.: Research Commentary–Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. Information Systems Research 21(4), 675–687 (2010), doi:10.1287/isre.1100.0323
13. Davenport, T.H.: The Future of Enterprise System-Enabled Organizations. Information Systems Frontiers 2(2), 163–180 (2000)
14. Ethiraj, S.K., Levinthal, D.: Modularity and Innovation in Complex Systems. Management Science 50(2), 159–173 (2004), doi:10.1287/mnsc.1030.0145
15. Schilling, M.A.: Toward a General Modular Systems Theory and its Application to Interfirm Product Modularity. Academy of Management Review 25(2), 312–334 (2000), doi:10.5465/AMR.2000.3312918
16. Simon, H.A.: The Architecture of Complexity. Proceedings of the American Philosophical Society 106(6), 467–482 (1962), doi:10.2307/985254
17. Boudreau, K.: Open Platform Strategies and Innovation: Granting Access vs. Devolving Control. Management Science 56(10), 1849–1872 (2010)
18. Baldwin, C.Y.: Where Do Transactions Come From? Modularity, Transactions, and the Boundaries of Firms. Industrial and Corporate Change 17(1), 155–195 (2008)

Software Ecosystem Roles Classification

Eko Handoyo[1,2], Slinger Jansen[1], and Sjaak Brinkkemper[1]

[1] Universiteit Utrecht, Utrecht 3584 CC The Netherlands
{eko.handoyo,slinger.jansen,s.brinkkemper}@uu.nl
[2] Universitas Diponegoro, Semarang 50275 Indonesia

Abstract. When studying software ecosystems, i.e., software-related organizations that collaboratively provide a market, it is difficult to identify the typical roles in that software ecosystem (SECO), and whether certain organizations belong to the ecosystem or not. The main aim of this research is to perform an independent literature review in order to create a classification of the typical roles named software ecosystem role classification (SERC). This research answered the research question, "How should a classification be formulated in order to provide the base for the software ecosystem roles?" The main result is a list of 5 major roles and 12 minor roles based on 9 papers that already discussed role identification in SECOs previously. The classification of these roles includes a description of the fundamental activities for each role. The classification enables those that model ecosystems and business models of software-related organizations to identify players quickly and their roles in ecosystems. Thereby furthering understanding of the underlying business models and value chains.

Keywords: software ecosystems, software industry, role classification, literature review.

1 Introduction

A software ecosystem (SECO) is defined as *"a set of organizations functioning as a unit and interacting with a shared market for software and services, together with the relationship among them"* [8]. SECO vision presents possible actors with the capability to observe moments and threats. SECO vision also approves actors to take on a role that influences the success of a SECO. In order to get a brief analysis of a SECO, several characteristics are already determined, i.e., composition of a SECO, entry barriers and stability. The composition of a SECO describes how it functions and how effectively it reacts to changes, i.e., what categories of actors occur, how broad they are, in what frequencies do they appear and what role do they take on [7]. The representatives of vendors are number, size and role. However, within such ecosystems, it is hard to differentiate these roles [4]. Presently, there exists a plethora of works describing the roles of actors in SECOs [15,14,10,13,3,11,6,12,1]. These works, however, do not claim to be comprehensive.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 212–216, 2013.

This paper develops a classification of SECO Roles. Such a classification helps SECO designers to identify typical roles in the SECOs. This led to the following research question, *How should a classification be formulated in order to provide the base for the software ecosystem roles?*

2 Role Classification

This study starts the role classification by reviewing two literature studies (i) a systematic mapping study on software ecosystems (SECOs), proposed by Barbosa and Alves [2], based on 44 papers (ii) a systematic literature review on SECOs, proposed by Manikas and Hansen [9], based on 90 papers.

Moreover, the second step of this study enhances the above-mentioned findings by doing an independent literature review. The literature review was directed at the topic of player roles in software ecosystems. The scope of the literature review was not limited exclusively to the literature published on software ecosystems, because of the robust analogy between the software industries and other business domains. To capture appropriate papers, books and articles, the following keywords were used: *software ecosystem, software ecosystems, software ecosystem roles, software ecosystem players, software ecosystem actors, software vendor, software supply industry, software producing organization.* The selection of these papers was rooted on the following various inclusion criteria (i) should discuss the business software ecosystem primarily (ii) should be written in English (iii) has been published and should be peer-reviewed (iv) has been already cited by other works (v) should be accessible (vi) should describe the role identification comprehensively. Regarding of the inclusion criteria, this study selects nine papers in established publications during the year 2000-2013.

2.1 Selected Papers

To facilitate discussions in the following sections, this section provide a short profile for each of the selected papers in the following paragraphs.

Digital Capital (DC). According to the book of *digital capital*, Tapscott et al. [15] differentiate between the following five classes of network participants: *customers, context providers, content providers, commerce service providers* and *infrastructure providers.*

Value Chain and Production Network (VN). In the paper of value chains and production networks, Sturgeon [14] mentions five types of network actors: *integrated firms, retailers, lead firms, turn-key suppliers* and *component suppliers.*

Software Value Chain (SC). In the book of *software ecosystem*, Messerschmitt and Szyperski [10] declare the decomposition of natural business functions consist of nine sources of values: *industry consultants, application software suppliers, infrastructure software suppliers, system integrators, infrastructure*

service providers, application service providers, information content suppliers, business consultants and *end-users organization.*

Business Webs (BW). According to the book of Steiner [13], he differentiates between two roles: *shapers* and *adapters.*

Web Service Ecosystems (WE). In the paper of Barros and Dumas [3], they describe the concept of web service ecosystems consist of five actors: *customers, providers, mediators, specialist intermediaries* and *brokers.*

Network Centric Innovation (NI). In the book of Nambisan and Sawhney [11], they propose three players: *architects, adapters* and *agents.*

Software Supply Network (SN). According to the paper of Jansen et al. [6], they recommend twelve lists of software supply network roles: *value-added resellers, resellers, software publishers, software designers, requirements engineers, software developers, product deployers, application service providers, independent software vendors, components-off-the-self vendors* and *original design manufacturers.*

Service Ecosystem (SE). Based on the paper of Riedl et al. [12], they state four network roles: *customers, platform providers, service providers* and *brokers.*

SaaS Ecosystem (Sa). A study of Abdat et al. [1] address the scope of a software ecosystem for five different key players: *SaaS vendors, SaaS providers, end users, resellers* and *integrators.*

2.2 Classification Development

This section describes the development of a new role classification, based on the nine selected papers that already exist in the previous sub-section. The development of a new role classification base on the following steps (i) list all the roles that already exist in the *selected papers* (ii) classify them into several general categories base on the activities, characteristics and the specific deliverables (iii) select the roles into the different-roles: unique elements (iv) develop a descriptive name for each category (v) classify the roles into major (bold) and minor roles. These steps propose a new role classification base on the two following principles (i) the roles should integrate and synthesize the earlier works in this domain (ii) the roles should be simple enough so that it can be easily understood, communicated and remembered. This section applies some symbols of Hong et al. [5], in order to classify the roles within the selected papers clearly: (i) the "=" symbol to indicate that the activities are the same (ii) the "<" or ">" symbols to indicate whether an activity in the major role comprises more or less than the activity in the concerning role, respectively (iii) the "><" symbol to indicate that the activity in the major role partly overlaps the activity of the concerning role (iv) in case a field in the *selected papers* is left blank it means that the activity is not present in the concerning role. Finally, this led to a software ecosystem roles classification (SERC), outlined in Table 1.

Table 1. Software Ecosystems Roles Classification

Role	Activity	Selected papers								
		DC	VN	SC	BW	WE	NI	SN	SE	Sa
Software vendors	**Evolve software**		<	>			<	<		
-Commercial-off-the-self vendors	Build & sell							=		
-Original-design manufacturers	Design, develop & sell							=		
-Platform\SaaS providers	Provide environment	>			<	<	<		=	=
-SaaS vendors	Provide software as a service									=
Service providers	**Contribute service**	=		=	<	<	<		=	
-Product distributors	Deploy, implement & resell							=		
-Software developers	Develop & supply							=		
-Software designers	Supply design							=		
-Application service providers	Supply computer service		>	=				=		
-Requirement engineer	Supply requirement document							=		
-Integrators	Customize user			=			>			=
-Content suppliers	Supply content			=						
Infrastructure providers	**Provide infrastructure**	=		=						
Resellers	**Buy & resell**		=			>		=	<	
-Value-added resellers	Add & resell							=		
Customers\End users	**Request service\product**	=		><		=			=	><

3 Conclusion

This paper develops a classification of software ecosystem (SECO) roles. In order to build it, this study conducted an independent literature review, a body of 9 papers discussing upon the identification of SECO roles. This research determined a list of 5 major roles and 12 minor roles. Furthermore, this study completed a description of the fundamental activities for each role. Due to the fast growing of SECOs domain in the software industry, SECO roles are essential concepts to uncover the actor's strategies to play within its SECO.

4 Further Research

First, the new role classification is currently not validated. The classification can be taken a step further by confirming with industry experts.

Second, the inclusion criteria that applied in the paper selection might be extended further, i.e., included the open source software ecosystems to be discussed entirely. According to Barbosa and Alves [2], they stated that such ecosystem is the most areas that published in the SECO domain. Therefore, it would be a significant point to examine such ecosystems.

Acknowledgments. This research project has been supported and financed by the Indonesian PhD scholarship grant number: 3374.1/E4.4/2011.

References

1. Abdat, N., Spruit, M., Bos, M.: Software as a service and the pricing strategy for vendors. AEBR Book Series, pp. 154–192 (2010)
2. Barbosa, O., Alves, C.: A systematic mapping study on software ecosystems. In: Proc of IWSECO (2011)
3. Barros, A.P., Dumas, M.: The rise of web service ecosystems. IT Professional 8(5), 31–37 (2006)
4. Burkard, C., Draisbach, T., Widjaja, T., Buxmann, P.: Software ecosystems: Vendor-sided characteristics of online marketplaces. In: Informatik (2011)
5. Hong, S., van den Goor, G., Brinkkemper, S.: A formal approach to the comparison of object-oriented analysis and design methodologies. In: Proc. of the 26th Hawaii Int'l Conf. on System Sciences, vol. 4, pp. 689–698. IEEE (1993)
6. Jansen, S., Brinkkemper, S., Finkelstein, A.: Component assembly mechanisms and relationship intimacy in a software supply network. In: 15th Int'l EurOMA Conference (2008)
7. Jansen, S., Brinkkemper, S., Finkelstein, A.: Business network management as a survival strategy: A tale of two software ecosystems. In: 1st IWSECO, vol. 505, pp. 34–48 (2009)
8. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: 31st Int'l Conf. on Software Engineering-Companion, pp. 187–190. IEEE (2009)
9. Manikas, K., Hansen, K.M.: Software ecosystems-a systematic literature review. Journal of Systems and Software (2012)
10. Messerschmitt, D.G., Szyperski, C.: Software ecosystem: understanding an indispensable technology and industry. MIT Press Books (2003)
11. Nambisan, S., Sawhney, M.S.: The global brain: your roadmap for innovating faster and smarter in a networked world. Wharton School Pub. (2008)
12. Riedl, C., Böhmann, T., Leimeister, J.M., Krcmar, H.: A framework for analysing service ecosystem capabilities to innovate. In: Proc. of 17th ECIS (2009)
13. Steiner, F.: Formation and early growth of business webs: modular product systems in network markets, vol. 8166. Physica-Verlag HD (2004)
14. Sturgeon, T.J.: How do we define value chains and production networks? IDS bulletin 32(3), 9–18 (2009)
15. Tapscott, D., Lowy, A., Ticoll, D.: Digital capital: Harnessing the power of business webs. Harvard Business Press (2000)

Formal Description for SaaS Undo

Hernán Merlino, Oscar Dieste, Patricia Pesado, and Ramón García-Martínez

PhD Program on Computer Sc. School of Computer Sc. National University of La Plata
Information Systems Research Group. Productive & Technologic Development Dept.
National University of Lanús
Instituto de Investigaciones en Informática LIDI. Facultad de Informática. UNLP - CIC
Empirical Software Eng. Group. School of Computer Sc. Madrid Polytechnic University
hmerlino@gmail.com, odieste@fi.upm.es,
ppesado@lidi.info.unlp.edu.ar, rgarcia@.unla.edu.ar

Abstract. This paper proposes a highly automated mechanism to build an undo facility into a new or existing system easily encapsulated into a service. The use of services strategy simplifies greatly the design of the undo process and encapsulates most of the functionalities required. We present a formal description when to use this service under alignments of software as a service.

Keywords: Undo, Services as a Service, and Usability component.

1 Introduction

Usability patterns were conceived with the aim of making usable software development simpler and more predictable [1]; in general usability requirements are included at an advanced stage of system development [2], when there is little time left and the key design decisions have already been taken.

The goal of this paper is to provide a formal description to detect availability to include software as a service (SaaS) [3] for undo usability patterns [4]. This provides the functionality necessary to undo actions taken by system users. This team decided to start with Undo pattern, because it is a common usability features in the literature [5].

Several authors have proposed alternatives to undo pattern, these alternatives focus on particular applications, notably document editors [6-7] although the underlying concepts are easily exportable to other domains. However, these proposals are defined at high level, without an implementation (or design) reusable in different types of systems. These proposals, therefore, do not solve the problem of introduction of usability features in software.

Undo has two alternatives of implementation in a system: (a) state operations. This option is present in systems where Undo functionality is a core for application, e.g. word editors, Applications without these functionality are not an option; (b) stateless operation. In these applications Undo functionality is only a plus for application, e.g. applications with forms to include and update data in a data base.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 217–222, 2013.

Our proposal detects a second subset of cases (stateless operations) in a highly efficient manner. If formal description is aligned to application or section of application, the architect can use SaaS for Undo [3]. The importance of having an automated solution for those is that they are the most frequents operations that occur in information systems.

The use of services for building applications is a very efficient way to reduce complexity and development time, creating an Undo service is a valid alternative to be taken into account by software engineering. We have implemented the framework to use Software as a Service (SaaS). Beyond scope of this article, the research team is working on the realization of a SOA model [8].

This article is structured as follows. Section 2 describes the state of the art regarding the implementation of undo. Section 3 presents the undo infrastructure, whereas, finally, Section 4 briefly discusses and presents the main contributions of our work.

2 Background

Undo is a very widespread feature, and is prominent across the whole range of graphical or textual editors, like, for example, word processors, spreadsheets, graphic editors, etc. Not unnaturally a lot of the undo-related work to date has focused on one or other of the above applications. For example, [6] Baker and Storisteanu [9] have patented two methods for implementing undo in document editors within single-user environments.

There are specific solutions for group text editors that support undo functionality, such as in Sun [10] and Chen and Sun [11] and Yang [12]. The most likely reason for the boom of work on undo in the context of document editors is its relative simplicity.

The problems of undo in multi-user environments have also attracted significant attention. Abrams and Oppenheim [13] have proposed mechanisms for using undo in distributed environments, and Abowd and Dix [4] proposed a formal framework for this field. In distributed environments, the solution has to deal with the complexity of updates to shared data (basically, a history file of changes) [14].

Several papers have provided insight on the internal aspects of undo, such as Mancini [15], who attempted to describe the undo process features.

Another important aspect which has been worked out is the method of representation of the actions performed by the users in Washizaki and Fukazawa [16], where a dynamic structure of commands is presented that represents the history of commands implemented.

Patents, like the method for building an undo and redo process into a system, have been registered [17]. Interestingly, this paper presents the opposite of an undo process, namely redo, which does again what the undo previously reverted. Other authors address the complexities of undo/redo as well. Thus, for example, Nakajima and Wash [18] define a mechanism for managing a multi-level undo/redo system.

The biggest problem with the above works is that, again, they are hard to adopt in software development processes outside the document editor domain. The only

noteworthy exception to this is a design-level mechanism called Memento [19]. Uses of Services in the enterprise build architecture models that are directly dependent upon the business strategy [20]. Service oriented architecture has the following characteristics [21]: (a) services are self-contained and modular (b) services support interoperability, (c) services are loosely coupled, (d) services are location-transparent, (e) services are composite modules, comprised of components.

The solutions presented are optimized for particular cases and are difficult to apply to other domains; on the other hand, it is necessary to include a lot of code associated with Undo in the host application.

3 Theoretical Justification

This will be done in two steps; first we will describe how to undo operations that do not depend on its state, the procedure to undo these operations consists in reinjection input data at time t-1, second we prove that reinjection input always produces correct results.

3.1 Initial Description

The most commonly used option for developing an undo process is to save the states of objects that are liable to undergo an undo process before they are put through any operation; this is the command that changes the value of any of their attributes. This method has an evident advantage; the system can revert without having to enact a special-purpose process; it is only necessary to remove and replace the current in-memory objects with objects previously saved.

This approach is a simple mechanism for implementing the undo process, although it has some weaknesses. On one hand, saving all the objects generates quite a heavy system workload. On the other hand, developers need to create explicit commands for all operations systems. Finally, the system interfaces (mainly the user interface) have to be synchronized with the application objects to enact an undo process. This is by no means easy to do in monolithic systems, but, in modern distributed computer systems, where applications are composed of multiple components all running in parallel (for example, J2EE technology-based EJB), the complications increase exponentially.

There is a second option for implementing an undo process. This is to store the operations performed by the system instead of the changes made to the objects by these operations. In this case, the undo would execute the inverse operations in reverse order. However, this strategy is seldom used for two reasons. On one hand, except for a few exceptions like the above word processing or spreadsheet software, applications are seldom designed as a set of operations. On the other hand, some operations do not have a well-defined inverse (imagine calculating the square of a table cell; the inverse square could be both a positive and a negative number).

The approach that we propose is based on this last strategy, albeit with a more simplified complexity. The key is that, in any software system whatsoever, the only

commands processed that are relevant to the undo process are the ones that update the model data (for example, a data entry in a field of a form that updates an object attribute, the entry of a backspace character that deletes a letter of a document object, etc.). In most cases, such updates are idempotent, that is, the effects of the entry do not depend on the state history. This applies to the form in the above example (but not, for example, to the word processor). When the updates are idempotent, neither states of the objects in the model or executed operations has to be stored, and the list of system inputs is only required.

3.2 Formal Description

The following definitions and propositions are used to prove (in an algebraic way) that UNDO process (UNDO transformation) may be built under certain process (transformation) domain constrains.

Definition 1. Let $E = \{ \varepsilon_j^i / \varepsilon_j$ is a data structure$\}$ be the set of all data structures.

Definition 2. Let ε_j^i be the instance i of data structure ε_j belonging to E.

Definition 3. Let $\varepsilon_j^C = \{ \varepsilon_j^i / \varepsilon_j^i$ is an instance i of the structure $\varepsilon_j\}$ be the set of all the possible instances of data structure ε_j.

Definition 4. Let $o_\tau^{\varepsilon j}$ be a transformation which verifies $o_\tau^{\varepsilon j} : \varepsilon_j^C \rightarrow \varepsilon_j^C$ and $o_\tau^{\varepsilon j}(\varepsilon_j^i) = \varepsilon_j^{i+1}$.

Definition 5. Let ε_j^{Cr} be a constraint of ε_j^C defined as $\varepsilon_j^{Cr} = \{ \varepsilon_j^i / \varepsilon_j^i$ is an instance i of the data structure ε_j which verifies $o_\tau^{\varepsilon j}(\varepsilon_j^{i-1}) = \varepsilon_j^i\}$

Proposition 1. If $o_\tau^{\varepsilon j} : \varepsilon_j^C \rightarrow \varepsilon_j^{Cr}$ then $o_\tau^{\varepsilon j}$ is bijective. Proof: $o_\tau^{\varepsilon j}$ is injective by definition 4, $o_\tau^{\varepsilon j}$ is surjective by definition 5, and then $o_\tau^{\varepsilon j}$ is bijective for being injective and surjective. QED.

Proposition 2. If $o_\tau^{\varepsilon j} : \varepsilon_j^C \rightarrow \varepsilon_j^{Cr}$ then has inverse. Proof: Let $o_\tau^{\varepsilon j}$ be bijective by proposition 1, then by usual algebraic properties $o_\tau^{\varepsilon j}$ has inverse. QED.

Definition 6. Let o_τ be the set of all transformations $o_\tau^{\varepsilon j}$.

Definition 7. Let Φ be the operation of composition defined as usual composition of algebraic transformations.

Definition 8. Let Σ be the service defined by structure $< E^\Sigma , o_\tau^\Sigma , \Phi >$ where $E^\Sigma \subseteq E$ and $o_\tau^\Sigma \subseteq o_\tau$.

Definition 9. Let $X = o_\tau^{\varepsilon j1} \Phi o_\tau^{\varepsilon j2} \Phi \ldots \Phi o_\tau^{\varepsilon jn}$ be a composition of transformations which verifies $o_\tau^{\varepsilon ji} : \varepsilon_j^C \rightarrow \varepsilon_j^{Cr}$ for all i:1...n. By algebraic construction $X : \varepsilon_j^C \rightarrow \varepsilon_j^{Cr}$.

Proposition 3. The composition of transformations X has inverse and is bijective. Proof: Let be $X = o_\tau^{\varepsilon j1} \Phi o_\tau^{\varepsilon j2} \Phi \ldots \Phi o_\tau^{\varepsilon jn}$. For all i:1...n verifies $o_\tau^{\varepsilon ji}$ has inverse by proposition 2. Let $[o_\tau^{\varepsilon ji}]^{-1}$ be the inverse transformation of $o_\tau^{\varepsilon ji}$, by usual algebraic properties $[o_\tau^{\varepsilon ji}]^{-1}$ is bijective. Then it is

possible to compose a transformation $X^{-1} = [o_\tau^{\varepsilon jn}]^{-1} \Phi [o_\tau^{\varepsilon jn-1}]^{-1} \Phi ... \Phi [o_\tau^{\varepsilon j1}]^{-1}$. The transformation X^{-1} is bijective by being composition of bijective transformations. Then transformation $X^{-1} : \varepsilon_j^{Cr} \rightarrow \varepsilon_j^{C}$ exists and is the inverse of X. QED.

Definition 10. Let UNDO be the X^{-1} transformation of X.

3.3 Use Method

If the evaluated system is aligned with the formal description detailed above, architect could use SaaS described in [4]; in another way, probably the architect needs to use any of specific domain's implementations of Undo detailed in section 2 (Background).

4 Conclusions

In this paper we have proposed a formal description to detect a sub set of Undo functionality and an alternative to implement this usability functionality in a system. The most salient feature of this framework is the type of information it stores to be able to undo the user operations: input data instead of in-memory object states or commands executed by the system. This lessens the impact of building the framework into the target application a great deal.

Building an Undo Service has some significant advantages with respect to Undo models presented. First of all the simplicity of inclusion in a host application under construction or existing, can be seen in the proof of concept. Second, the independence of service in relation to the host application allows the same architectural model to provide answers to different applications in different domains. Construction of a service allows Undo to be a complex application, with the possibility of including analysis for process improvement, as described in the next paragraph it is possible to detect patterns of invocation of Undo in different applications.

Further work is going to bring: (a) creation of a pre-compiler, (b) automatic detection of fields to store (c) extension of the framework to other platforms.

Acknowledgements. The research reported in this paper has been partially funded by grants UNLa-SCyT-33A167 and UNLa-SCyT-33B112 of the National University of Lanus (Argentine) and by grants TIN2008-00555 and HD2008-00046 of the Spanish Ministry of Science and Innovation (Spain).

References

1. Ferre, X., Juristo, N., Moreno, A.: Framework for Integrating Usability Practices into the Software Process. Madrid Polit. University (2004)
2. Ferre, X., Juristo, N., Moreno, A., Sanchez, I.: A Software Architectural View of Usability Patterns. In: 2nd Workshop on Software and Usability Cross-Pollination (INTERACT 2003), Zurich, Switzerland (2003)

3. Merlino, H., Dieste, O., Pesado, P., García-Martínez, R.: Service Oriented Architecture for Undo Functionality. In: Proceedings 6th International Conference on Research and Practical Issues of Enterprise Information Systems (2012)
4. Merlino, H., Dieste, O., Pesado, H., García-Martínez, R.: Software as a Service: Undo. In: Proceedings 24th International Conference on Software Engineering and Knowledge Engineering (SEKE 2012), pp. 328–332 (2012) ISBN 978-1-891706-31-8
5. Abowd, G., Dix, A.: Giving UNDO attention. University of York (1991)
6. Qin, X., Sun, C.: Efficient Recovery algorithm in Real-Time and Fault-Tolerant Collaborative Editing Systems. School of computing and Information Technology Griffith University Australia (2001)
7. Bates, C., Ryan, M.: Method and system for UNDOing edits with selected portion of electronic documents. PN: 6.108.668 US (2000)
8. Merlino, H., Pesado, P., Dieste, O., García-Martínez, R.: Inclusion Process of UNDO/REDO Service in Host Applications. In: Software Engineering, Methods, Modeling and Teaching, Edited by Pontificia Universidad Católica de Peru. JIISIC 2012, Lima, Peru, vol. II (2011)
9. Baker, B., Storisteanu, A.: Text edits system with enhanced UNDO user interface. PN: 6.185.591 US (2001)
10. Sun, C.: Undo any operation at time in group editors. School of Computing and Information Technology, Griffith University Australia (2000)
11. Chen, D., Sun, C.: Undoing Any Operation in Collaborative Graphics Editing Systems. School of Computing and Information Technology, Griffith University Australia (2001)
12. Yang, J., Gu, N., Wu, X.: A Documento mark Based Method Supporting Group Undo. Department of Computing and Information Technology, Fudan University (2004)
13. Abrams, S., Oppenheim, D.: Method and apparatus for combining UNDO and redo contexts in a distributed access environment. PN: 6.192.378 US (2001)
14. Berlage, T., Genau, A.: From Undo to Multi-User Applications. German National Research Center for Computer Science (1993)
15. Mancini, R., Dix, A., Levialdi, S.: Reflections on UNDO. University of Rome (1996)
16. Washizaki, H., Fukazawa, Y.: Dynamic Hierarchical Undo Facility in a Fine-Grained Component Environment. Department of Information and Computer Science. Waswda University, Japan (2002)
17. Keane, P., Mitchell, K.: Method of and system for providing application programs with an UNDO/redo function. PN:5.481.710 US (1996)
18. Nakajima, S., Wash, B.: Multiple levels UNDO/redo mechanism. PN: 5.659.747 US (1997)
19. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison- Wesley (1994)
20. Binildas, C.A., Malhar, B., Vincenzo, C.: Service Oriented Architecture with Java. Packt Publishing, Birmingham – Mumbai (2008)
21. Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., Luo, L., Newling, T.: Patterns: Service-Oriented Architecture and Web Services. IBM, Redbooks (2004)

Virtual Character Based Interactive Interfaces for Deaf and Functionally Illiterate Users

Nadeem Ahmad

Department of Control and Computer Engineering
Politecnico di Torino, Italy
nadeem.ahmad@polito.it
http://www.testgroup.polito.it

Abstract. Availability of technology does not ensure accessibility. The heavy use of text on everything from menus to document content means that those who are deaf or functionally illiterate are not able to access functions and services implemented on most computer software. The research is motivated by objective to provide applications with easy to use interfaces for communities of deaf and illiterate users, which enables them to work without any human assistance.

Keywords: ICT4D, Interfaces, Functionally Illiterate, Deaf, Assistive Technologies.

1 Virtual Character Based Interfaces

Illiteracy estimates show that in 2008, 796 million adults worldwide (15 years and older) were not being able to read and write and two-thirds of them (64%) were women [1]. On the other hand there are 60 thousand deaf users only in Italy (Feb 2010), 3 Million in France (May 2010) and 1 Million in Germany (April 2010) [2]. Interfaces between technology and society need to be different, as level of understanding of users is very different. We are considering two type of users, deaf and functionally illiterate, although they perceive in different ways but designing interfaces for functionally illiterate persons may in some respects resemble designing for people who are cognitively challenged, since some of their cognitive abilities may be less developed than those of literate people. There is no such thing as a one size fits-for-all for this type of project. Sensitivity to context and diversity are key factors to concentrate. The idea to use Virtual Character is not new for us; but mostly Virtual characters were used as role player in virtual environment, rarely interfaces are designed in which virtual avatars spoke local regional language and guided the end user that how to perform his task. Mr. Clippy introduced in Office 97 was impolite but the major reason behind its failure was the ability to take preemptive control of user's cursor which is not in our case. Avatars are natural candidates for the development of sign language and it is established fact that virtual character based interfaces are useful for functionally illiterate users in rapid learning [3]. We developed a Virtual Character based Italian Sign Language Dictionary to support Deaf learning of

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 223–224, 2013.

both Sign Language and written language. It provides full set of lexemes and videos data set and an online interface with MultiWordNet synsets [4]. The dictionary allows users to acquire information about lemmas, synonyms and synsets in the Sign Language. The application is platform independent and can be used on any operating system [5,6]. Figure 1 shows abstract level details to add virtual character in applications exploiting user interfaces. The raw animation files and rendering engine to produce virtual character based motion frames are part of Virtual Character Space and will act like a shell. Virtual characters are source to enhance deaf learning by translating written contents into sign language and external human assistance is being substituted by these life like characters for functionally illiterate users.

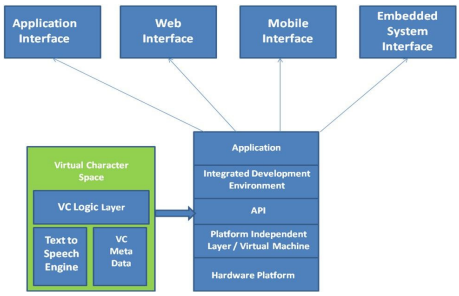

Fig. 1. Inclusion of Virtual Characters in real applications

References

1. UIS Fact Sheet: Adult and youth literacy: Global trends in gender parity. UIS Fact Sheet 3, UNESCO Institute for Statistics, Montreal, Canada (September 2010)
2. Galaudet University Library: Deaf Statistics: Deaf populations overseas (June 2010)
3. Toyama, K., Sagar, A., Medhi, I.: Computer interface for illiterate and near-illiterate users. US Patent 7,603,621 (October 13, 2009)
4. Ahmad, N., Barberis, D., Garazzino, N., Prinetto, P., Shoaib, U., Tiotto, G.: A virtual character based italian sign language dictionary. In: Proceedings of the 13th International Conference on Computers Helping People with Special Needs, Linz, Austria (July 2012)
5. Shoaib, U., Ahmad, N., Prinetto, P., Tiotto, G.: A platform-independent user-friendly dictionary from italian to lis. In: Chair, N.C.C., Choukri, K., Declerck, T., Doğan, M.U., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S. (eds.) Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012), European Language Resources Association (ELRA), Istanbul (May 2012)
6. Barberis, D., Garazzino, N., Prinetto, P., Tiotto, G., Savino, A., Shoaib, U., Ahmad, N.: Language resources for computer assisted translation from italian to italian sign language of deaf people. In: Proceedings of Accessibility Reaching Everywhere AEGIS Workshop and International Conference, Brussels, Belgium (November 2011)

Simplicity in Application Development for Business Model Design

Steve Boßelmann

Department of Computer Science
University of Potsdam
August-Bebel-Str. 89
14482 Potsdam, Germany
bossel@cs.uni-potsdam.de

Abstract. Discussing business models of companies and organizations has increased in recent years, particularly from an economics perspective. And so has the interest in creating graphical representations that emphasize essential factors. However, feasible implementations of modeling tools are rare, as they tend to be domain specific but at the same time tailored towards the requirements of a heterogeneous group of stakeholders. As ideally the latter should be encouraged to express their needs in a simple but meaningful way, the author researches the application of the eXtreme Model Driven Design (XMDD) approach to model critical parts of an application in order to keep it as simple as possible for participants to contribute to the development process.

Keywords: business model design, model-driven software development, simplicity.

1 Summary

Tools like the Business Model Canvas (BMC) based on the work by Osterwalder [1] receive a lot of attention in the area of business economics. However, the term 'tool' is ambiguous and in this case should be considered from a business economics perspective, i.e. a best practice approach on gathering the most important components of a business model by asking the right questions. However, from the computer science perspective there are virtually no sophisticated tools that support business model design by means of applying some kind of formal semantics, neither in general nor for specific areas of application.

Creating such a tool is challenging for numerous reasons. On the one hand there is neither common sense about a suitable ontology nor about component types that business models should actually comprise. Hence particular model characteristics depend on the application area as well as the actual business domain. On the other hand, the development process depends on inter-disciplinary collaboration and communication. It requires immediate contribution of a heterogeneous set of stakeholders involving business managers as well as economics researchers, most of them lacking broad knowledge of formal models and software engineering skills.

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 225–226, 2013.

Putting it all together, a decent amount of simplicity throughout the development lifecycle is key for success. Recently, the notion of simplicity as a driving paradigm in information system development has been explicitly identified as an important research topic, yet poorly understood [2]. The author leverages the jABC Framework [3] and the advantageous characteristics of its architecture following the XMDD approach [4] to model critical parts of an application in a graphical and domain-specific process notation in order to keep it as simple as possible for participants to contribute to the development process. He researches practicability, benefits and drawbacks of this approach based on an exemplary case study in the healthcare domain by means of the development of a tool for creating innovative business models of diagnostic companies [5].

References

1. Osterwalder, A.: The Business Model Ontology - A Proposition in a Design Science Approach. Dissertation (2004)
2. Margaria, T., Steffen, B.: Simplicity as a Driver for Agile Innovation. Computer 43(6), 90–92 (2010)
3. Steffen, B., Margaria, T., Nagel, R., Jörges, S., Kubczak, C.: Model-Driven Development with the jABC. In: Bin, E., Ziv, A., Ur, S. (eds.) HVC 2006. LNCS, vol. 4383, pp. 92–108. Springer, Heidelberg (2007)
4. Margaria, T., Steffen, B.: Service-Orientation: Conquering Complexity with XMDD. In: Hinchey, M., Coyle, L. (eds.) Conquering Complexity. Springer (2012)
5. Project, Service Opportunities in Personalized Medicine, `http://dpm.ceip.de`

Software Ecosystem Modeling

Eko Handoyo[1,2]

[1] Universiteit Utrecht, Utrecht 3584 CC, The Netherlands
eko.handoyo@uu.nl
[2] Universitas Diponegoro, Semarang 50275 Indonesia

Abstract. Increasingly, software producing organizations and their partners are organizing around extendable platforms, forming software ecosystems, a topic that is receiving increasing attention at the moment. Unfortunately, inadequate modeling guidelines exist currently for creating theoretical and visual models of software ecosystems, hampering advancements in this area. Even though, many different tools are available, these do not provide pointers and guidelines for performing correct and insightful modeling of software ecosystems. The aim of this PhD project is to create a modeling framework, consisting of tools, guidelines, and foundations for modeling software ecosystems. With such a framework, software ecosystem researchers, policy makers, and strategic managers of software producing organizations will gain insight into their ecosystems quicker and with higher fidelity.

Keywords: Software ecosystems, modeling, open source, role classifications, business modeling.

One of the challenges identified in the research agenda of Jansen et al. [4] is the modeling of software ecosystems (SECOs). Although a plethora of modeling tools and methods can be found, none of them provide a uniform way to model SECOs in an insightful, comparative, efficient, and visually appealing manner. The aim of this PhD thesis is to formalize models of SECOs and their modeling processes and enable SECO researchers to provide insightful models. In the work on software business modeling of Lucassen et al. [5], we compare three visual business modeling techniques with a visual approach and identified the strong and weak points of each modeling technique, based on applying the techniques to three start-ups and interviews with industry experts. One of the findings of this research and the research agenda of Jansen et al. is that SECO modeling still resides in an early stage. The research project consists of the following steps:

1. **Identify weaknesses in current modeling methods -** By performing a structured literature survey and reusing literature surveys in the domain, we are able to identify limitations of current modeling methods. The preliminary conclusions of this are already presented in the study of Lucassen et al. [5]
2. **Establish roles in SECOs -** As there seems to exist a multitude of roles in typical SECOs, from service providers to open source contributors, and from keystone organizations like Microsoft to app builders like Rovio games, a formal role classification needs to be established that enables one to identify

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 227–228, 2013.

all roles in a SECO. The fundamental results of this are formerly published in the paper of Handoyo et al. [3].

3. **Perform evaluation of modeling tools -** As the literature illustrates typical modeling methods, we also do a tool evaluation, in which tools are compared on their ability to model large-sized ecosystems in an insightful way. Also, descriptions are identified to provide a framework with typical characteristics that can be collected about a SECO.
4. **Formalize modeling for commercial SECOs -** The descriptions, roles, and models are combined into a formal modeling method that enables any SECO modeler to quickly create insightful, useful, and effective models of SECOs in a commercial context.
5. **Generalize modeling method to open source SECOs -** If possible the method is expanded to include open source ecosystems as well, as these introduce the concepts of one-man entities, foundations, donations, etc.

Finally, the importance of this project is to establish the SECOs formal characteristics. The purpose is to serve as a foundation in order to further analyze the complete SECO and answer the main Research Question (RQ): *How can a modeling framework be created that establishes a method for SECO model building that is effective, uniform, efficient, and useful?* The scientific contribution of this work will be a formal method for modeling a software ecosystem and potentially also contributes to the domain of business ecosystem modeling. The societal contribution will be a method that can be used to quickly develop a SECO model for any ecosystem that is insightful, effective, efficient, and useful in a commercial or open source context. Finally, the method provides strategists with concrete tools to gain insight into SECOs and possibly alter their SECOs based on the models.

Acknowledgments. This research project has been supported and financed by the Indonesian PhD scholarship grant number: 3374.1/E4.4/2011.

References

1. Barbosa, O., Alves, C.: A systematic mapping study on software ecosystems. In: Proc of IWSECO (2011)
2. Boucharas, V., Jansen, S., Brinkkemper, S.: Formalizing software ecosystem modeling. In: Proc. of the 1st int'l Workshop on Open Component Ecosystems, pp. 41–50. ACM (2009)
3. Handoyo, E., Jansen, S., Brinkkemper, S.: Software ecosystem roles classification. In: Proc. of the Int'l Conf. on Software Business (2013)
4. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: 31st Int'l Conference on Software Engineering-Companion Volume, ICSE-Companion 2009., pp. 187–190. IEEE (2009)
5. Lucassen, G., Brinkkemper, S., Jansen, S., Handoyo, E.: Comparison of visual business modeling techniques for software companies. In: Cusumano, M.A., Iyer, B., Venkatraman, N. (eds.) ICSOB 2012. LNBIP, vol. 114, pp. 79–93. Springer, Heidelberg (2012)
6. Manikas, K., Hansen, K.M.: Software ecosystems-a systematic literature review. Journal of Systems and Software (2012)

Impact of Enterprise System Modularity on Process Performance

Carl Simon Heckmann

Institute for Enterprise Systems, University of Mannheim,
68131 Mannheim, Germany
heckmann@es.uni-mannheim.de

Abstract. Modularization and platform-based product development has been adopted in a variety of engineering disciplines. In the information system context modularity has been proposed as a means for increasing flexibility-to-change and process performance but no evidence has been provided and limited further analysis of this link could be observed. This Ph.D. project provides an empirical analysis of the relationship between the degree of modularity of an enterprise system and the performance of the supported business processes.

Keywords: Enterprise System, Platform, Modularization, Product Development, Flexibility-to-change, Business Process Performance.

1 Research Abstract

Modularity as a concept received widespread attention in a variety of disciplines, reaching from product [1] to organizational design [2], and various industries such as automotive, aircraft or consumer electronics [3]. Following modular systems theory a modular architecture is associated with a huge range of benefits such as increased flexibility, reduced product- and switching costs [4] and improved innovation outcomes [5]. In the information system (IS) context modularity can increase flexibility to change a system and business process efficiency [6, 7]. However, no guidelines about the optimal degrees of modularity are provided.

So far modularity has received little analysis in the context of enterprise systems (ES). ES can be conceptualized as socio-technical systems [8, 9], consisting of a core platform and a flexible sphere on both the organizational and technological level. This study focusses on the application system only, typically building on packaged software such as enterprise resource planning (ERP). However there is limited guidance on the degree of modularity suited for clearly allocating modules to either the core platform or the flexible sphere. The impact of IT on the enterprise level can only be measured through intermediate contributions on the process level [10]. Consequently the effect of modularity of the application system on process performance will be analyzed in this study. Therefore this Ph.D. project is intended to answer the following research question:

What is the effect of application system modularity on process performance and how is this effect influenced by different integration requirements of modules?

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 229–230, 2013.

To answer this question a hybrid research approach will be applied. First hypotheses will be formulated on the basis of state-of-the-art literature and evaluated through case studies to build a model, which provides the base for a confirmatory quantitative study. The challenge how to achieve process efficiency while keeping the system flexible to change is relevant in all highly dynamic industries. As one instance of such an industry the German energy sector will provide the context for this study. This research focusses on the energy-supplier switching process as a single process to stay in a feasible scope. This process is selected as it is standardized [11] and a high-volume process, making it highly relevant for companies in this industry.

The contribution is to explain the link between the degree of modularity of an ES and process performance and to empirically confirm the proposed link between modularization and increased flexibility-to-change of the ES [7]. Thus this research contributes to modular systems theory by applying it in the context of ES [4]. Practitioners can benefit from guidelines which degrees of modularity of ES are optimal in different settings.

References

1. Jose, A., Tollenaere, M.: Modular and platform methods for product family design: literature analysis. Journal of Intelligent Manufacturing 16(3), 373–390 (2005), doi:10.1007/s10845-005-7030-7
2. Langlois, R.N.: Modularity in technology and organization. Journal of Economic Behavior & Organization 49(1), 19–37 (2002), doi:10.1016/S0167-2681(02)00056-2
3. Sanchez, R., Mahoney, J.T.: Modularity, Flexibility, and Knowledge Management in Product and Organization Design. Strategic Management Journal 17, 63–76 (1996), doi:10.2307/2486991
4. Schilling, M.A.: Toward a General Modular Systems Theory and its Application to Interfirm Product Modularity. Academy of Management Review 25(2), 312–334 (2000), doi:10.5465/AMR.2000.3312918
5. Ethiraj, S.K., Levinthal, D.: Modularity and Innovation in Complex Systems. Management Science 50(2), 159–173 (2004), doi:10.1287/mnsc.1030.0145
6. Gebauer, J., Schober, F.: Information System Flexibility and the Cost Efficiency of Business Processes. Journal of the Association for Information Systems 7(3), 122–146 (2006)
7. Gebauer, J., Lee, F.: Enterprise System Flexibility and Implementation Strategies: Aligning Theory with Evidence from a Case Study. Information Systems Management 25(1), 71–82 (2008), doi:10.1080/10580530701777198
8. Orlikowski, W.J.: The sociomateriality of organisational life: considering technology in management research. Cambridge Journal of Economics 34(1), 125–141 (2010), doi:10.1093/cje
9. Maedche, A., Mueller, B.: Enterprise Systems-A Research Agenda. SSRN 2033145 (2012)
10. Ray, G., Muhanna, W.A., Barney, J.B.: Information Technology and the Performance of the Customer Service Process: A Resource-Based Analysis. MIS Quarterly 29(4), 625–652 (2005), doi:10.2307/25148703
11. Bundesnetzagentur, Anlage zum Beschluss BK6-06-009 - Darstellung der Geschäftsprozesse zur Anbahnung und Abwicklung der Netznutzung bei der Belieferung von Kunden mit Elektrizität. GPKE (2011)

Managing Speed in Companies Developing Large-Scale Embedded Systems

Antonio Martini

Software Engineering Division
Chalmers University of Technology
Göteborg, Sweden
antonio.martini@chalmers.se

Abstract. An open issue is how to reach quickness and responsiveness in addressing customer needs within large-scale embedded system product development, where the processes are bound to the physical product development. Speed is a key quality that needs particular attention. We are developing a framework to understand what kinds of speed are important, what factors are determining them, what are the visible effects and what is possible to improve in order to reach speed related business goals.

Keywords: Agile software development, development speed, large scale software engineering, embedded systems, software business.

1 Contribution

Many industries in pure software development have recently adopted Agile Software Development in order to deliver customer values as fast as possible [1], [3]. An open issue is how to reach such quickness and responsiveness within large-scale embedded system product development, where the processes are bound to the physical product development [2]. A key quality of such process is the speed with which the software is rolled out. A company that seeks to optimize its return of investment of R&D (ROI of R&D) must manage three kinds of speed [4]:

- The speed with which customer needs lead to new product offers (*1st Deployment speed*),
- The speed with which new features are replicated in new products (*Replication speed*), and
- The speed with which change requests to an existing product are realized (*Evolution speed*).

In [5] we showed how these three kinds of speed depend on *Interaction speed*: how fast teams (or other organizational units), resolve each others' needs.

Different kinds of speed depend on a number of organizational, architectural, and individual factors that may or may not be managed (Fig. 1). The aim of my PhD thesis is to find such factors, to recognize the effects and to propose solutions (practices) to manage the factors, which in the end would help reaching speed-related business goals (and therefore optimizing Return of Investment of R&D).

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 231–232, 2013.

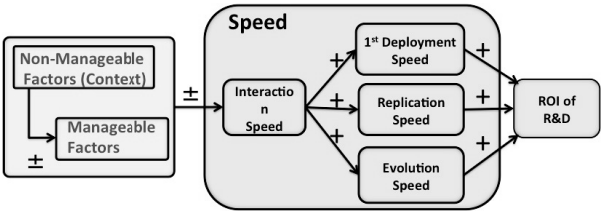

Fig. 1.

Our approach is to equip managers and teams with both theoretical and practical instruments for continuous process improvement (Fig. 2).

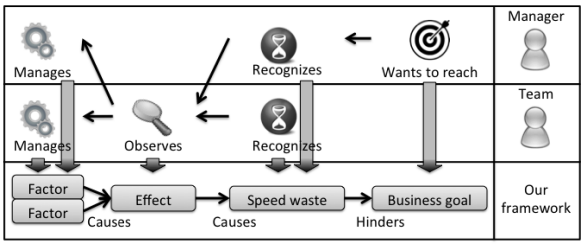

Fig. 2.

Speed wastes threaten the achievement of business goals, and can be recognized by visible effects, observable by managers together with the team(s). Effects, in turn are caused by one or more factors. Managers can investigate the status of such factors in the teams to find which one is the cause for the effect, reducing the solution space, saving time and resources. When the factors are recognized, both the team and the manager (depending on the factor) may decide to apply recommendations.

We have conducted multiple-case case studies in 3 large companies developing embedded software: so far, the 17 interviews and the survey of 35 respondents have highlighted different factors related to the mentioned kinds of speed: we have explored factors influencing 1st deployment speed, replication speed [4] and interaction speed [5]. An important set of factors is related to communication in ASD. We have proposed some practical guidelines that need further research and validation.

References

1. Bosch, J., Bosch-Sijtsema, P.M.: Introducing agile customer-centered development in a legacy software product line. Software: Practice and Experience 41 (2011)
2. Karlstrom, D., Runeson, P.: Combining agile methods with stage-gate project management. IEEE Software 22, 43–49 (2005)
3. Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., Kahkonen, T.: Agile software development in large organizations. Computer 37 (2004)
4. Martini, A., Pareto, L., Bosch, J.: Enablers and inhibitors for speed with reuse. In: Proceedings of the 16th International Software Product Line Conference - SPLC 2012. ACM (2012)
5. Martini, A., Pareto, L., Bosch, J.: Improving Businesses Success by Managing Interactions among Agile Teams in Large Organizations. Accepted for Publication in the 4th International Conference in Software Business (2013)

Global Manufacturing Networks as Software-Intensive Service Providers Motivation, Relevance, Research Objective

Tobias Tauterat

Graduate School of Advanced Manufacturing Engineering (GSaME)
Universität Stuttgart, Chair of Information Systems II (Business Software),
Keplerstraße 17, 70174 Stuttgart, Germany
Tobias.Tauterat@gsame.uni-stuttgart.de

Abstract. Developing the factory of the future is an objective of the Graduate School of Excellence advanced Manufacturing Engineering (GSaME) in Stuttgart. To achieve this goal several research areas concerning the factory of the future exist. For example, strategies and factories development, information and communication for manufacturing, material and process engineering, or management of global manufacturing networks. One specific research project within the area "management of global manufacturing networks" is "global manufacturing networks as software-intensive service providers", which started in December 2012. This research project deals with the questions, how the factory of the future is able to satisfy its increasing need of information and communication technology (ICT) in the future, and how to compose different ICT-services, which should be integrated in the existing global manufacturing network in a second step. The contribution in hand considers the motivation, relevance, and research objective of this research project.

Keywords: Global manufacturing networks, information and communication technology, manufacturing, service provider, software-intensive service provider.

1 Motivation

Against the background of global megatrends such as globalization, individualism, knowledge in global ICT, or urbanization, the manufacturing industry is facing new challenges in the future. To meet these challenges in order to develop the factory of the future several generic concepts already exist. For example, there is a need to react flexible on economic turbulences. So changes from linear and streamlined supply chains to global manufacturing networks can be observed. [1]

ICT has a twofold role in developing the factory of the future. On the one hand ICT can be considered as an enabler to design the factory of the future, such as building and supporting manufacturing ecosystems and their processes. On the other hand, using ICT has an influence on the design of the product itself. In addition to the still significant and undoubted manufacturing of physical goods, the net product potentialities will be fully exhausted in the course of industrial tertiarization by the

G. Herzwurm and T. Margaria (Eds.): ICSOB 2013, LNBIP 150, pp. 233–234, 2013.

integration of products and services to so called hybrid bundle of services in the future. The traditional factory using on-premise ICT will be extended by using and providing use-based ICT-services and architectures with high usability ("plug and produce") in the future. Manufacturing factories providing scalable hardware, software and service products will act as so called software-intensive service providers. [2]

2 Relevance

The relevance of this research project can be seen in two main facts:

1. The future vision of factories as software-intensive service providers is widely unexplored.
2. The integration of these software-intensive factories of the future within existing global manufacturing networks is widely unexplored as well.

If we take a detailed look at these two main facts, the role of ICT and its creation of value for global manufacturing networks are mostly unknown and derived business models are missing. Moreover there is a lack of knowledge about design rules for the systematic integration of software-intensive service providers into (existing) global manufacturing networks. This research project can help to close this gap by developing solutions for the vision of a global and adaptable manufacturing network as software-intensive and value-adding service provider.

3 Research Objective

Main objective of this research project is to answer the question, which contribution ICT can provide with regard to different design areas within a manufacturing factory as well as within a global manufacturing network in context of process, product and service design. Doing so, software-based business models and business model components, such as value proposition, value dissemination and value capture, have to be analyzed and designed. In addition it is necessary to explore how a possible software ecosystem has to be conceptualized for this new kind of factory. For example it has to be considered whether and how this software ecosystem can be integrated in the existing manufacturing network. During the design and the development of the software ecosystem it is necessary to consider the area of tension between standardization and customization of hybrid bundle of services, such as e-services and business software.

References

1. Westkämper, E.: Next Generation Manufacturing - Manufacturing 2030. In: Manufuture 2011 Conference, Wroclaw (2011)
2. Herzwurm, G., Pietsch, W., Schockert, S., Tauterat, T.: QFD for Cloud Computing. In: Proceedings of the 18th International Symposium on Quality Function Deployment, Tokyo (2012)

Author Index

Abrahamsson, Pekka 141, 166
Ahmad, Nadeem 223

Bloemendal, Ewoud 195
Bosch, Jan 60, 79
Boßelmann, Steve 225
Brinkkemper, Sjaak 212
Buxmann, Peter 1

Dieste, Oscar 217

Furtmueller, Elfi 26, 73

García-Martínez, Ramón 217
Gencel, Cigdem 166
Guceri-Ucar, Gozem 178

Handoyo, Eko 212, 227
Heckmann, Carl Simon 207, 229
Hess, Thomas 153
Hoerndlein, Christian 153
Holmström Olsson, Helena 79

Jansen, Slinger 183, 195, 212

Kaltenecker, Natalie 153
Koch, Stefan 178
Kude, Thomas 130

Laatikainen, Gabriella 117
Lucassen, Garm 183

Maedche, Alexander 207
Marciuska, Sarunas 166
Martini, Antonio 60, 231
Mazhelis, Oleksiy 117
Merlino, Hernán 217

Neubauer, Johannes 13
Novelli, Francesco 31

Ojala, Arto 117

Pareto, Lars 60
Pesado, Patricia 217
Phaphoom, Nattakarn 141
Pietsch, Wolfram 102
Popp, Karl Michael 130
Pussep, Anton 1

Riehle, Dirk 90
Runeson, Per 48

Schief, Markus 1
Schütz, Sebastian Walter 130
Steffen, Bernhard 13

Tate, Mary 26
Tauterat, Tobias 233

van Rooij, Kevin 183

Wagner, Marcus 43
Wang, Xiaofeng 141
Weikert, Florian 90
Wnuk, Krzysztof 48