7-15-2012

Software-As-A-Service Development: Driving Forces Of Process Change

Stuckenberg Sebastian
University of Mannheim, Schloss, 68131 Mannheim, Germany, stuckenberg@uni-mannheim.de

Beiermeister Stefan
Senacor Technologies AG, Düsseldorfer Strasse 13, 65760 Eschborn, Germany, stefan.beiermeister@senacor.com

SOFTWARE-AS-A-SERVICE DEVELOPMENT: DRIVING FORCES OF PROCESS CHANGE

Stuckenberg, Sebastian, University of Mannheim, Schloss, 68131 Mannheim, Germany, stuckenberg@uni-mannheim.de

Beiermeister, Stefan, Senacor Technologies AG, Düsseldorfer Strasse 13, 65760 Eschborn, Germany, stefan.beiermeister@senacor.com

Abstract

In the recent years, Software-as-a-Service has gained growing attention from software vendors as well as software customers. In this distribution and pricing model, software vendors are responsible for the operation and maintenance of solutions and customers pay for the service in form of continuous usage-based subscription fees. This study analyses the key characteristics of the Software-as-a-Service concept and evaluates their influence on development processes of software vendors. Based on the literature, two defining characteristics (vendor-hosted, pay-per-usage) and five supportive characteristics (standardization, web-technologies, multi-tenancy, fine-granularity, continuous evolution) are identified. Two vendors of complex business applications – both applying a mix of deployment models – are analyzed to identify the impact of Software-as-a-Service on development processes, as well as the driving forces behind the change.

The results indicate, that the concept especially affects the requirements engineering as well as operations phases. The combination of the defining characteristics results in an increased cost as well as innovation pressure. Vendors are challenged to optimize and streamline existing practices to reduce internal costs. At the same time, vendors need to innovate quicker. The combination of these aspects asks for an increased integration of internal activities like development and operations, as well as an increased customer-orientation and integration.

Keywords: Software-as-a-Service, impact, development process, methodology.

1 INTRODUCTION

Over the recent years, software vendors have introduced an increasing number of software solutions that apply the so called *Software-as-a-Service* concept. This delivery and pricing model envisions vendors to operate and maintain the software solution and supply these to customers via the internet charging continuous usage-based subscription fees (Sääksjärvi et al. 2005; Xin and Levina 2008). The concept is part of the Cloud-Computing paradigm and forms the software layer in the stack model above Platform-as-a-Service and Infrastructure-as-a-Service (Weinhardt et al. 2009). Similar to the overarching Cloud-Computing concept, the market for software service solutions is expected to experience continuous growth rates over the next years (Gartner 2009; West et al. 2010). The concept, therefore, is gaining relevance for research and practice.

Linked to the introduction of the concept are a number of open questions and challenges. Research and observations in the market suggest the concept to impact existing practices of software vendors (Juell-Skielse and Enquist 2012; Stuckenberg et al. 2011). Established vendors, particularly, seem to have difficulties in implementing the Software-as-a-Service model successfully. For instance, vendors have problems to reach the necessary cost level to operate their solutions profitable. Aspects like the total cost of ownership gain importance and bear unforeseeable problems that have already led in significant delays in the market introduction of an established vendor's Software-as-a-Service solution (Dignan 2008).

Before it is possible to propose appropriate practices to successfully deliver and operate Software-as-a-Service solutions, it is necessary to analyse the concept itself, the potential impacts on existing processes and structures and the driving forces behind these. Therefore, the aim of this research is to empirically analyse the potential impact of the Software-as-a-Service concept on the software development processes and to identify the concept's aspects that drive the need to adjust and change existing practises. The research questions can be summarized as:

How does Software-as-a-Service influence software development processes and what are the driving forces that require process changes?

The remaining sections of this paper are organized as follows: Section 2 provides a definition of the Software-as-a-Service concept and its key characteristics and reviews the literature for research discussing or proposing development processes for Software-as-a-Service solutions. Section 3 introduces the conducted case studies and the research methodology, followed by key findings of the cases in section 4. The paper ends with a discussion in section 5 and the conclusion.

2 LITERATURE REVIEW

To be able to evaluate the impact of the Software-as-a-Service concept on development and operation processes, it is important to clearly define the concept itself and its characteristics. The following section will therefore provide a definition of the concept and discuss some characteristics that are most often linked to Software-as-a-Service. Afterwards the current literature discussing the impact and proposed development methodologies are introduced.

2.1 Software-as-a-Service

In general, Software-as-a-Service can be seen as a software distribution and pricing model of software vendors. It mainly distinguishes itself from traditional models by applying a different approach of software delivery to the end user and a changed licensing model, as well as the application of various new technologies. The software solutions are operated and maintained by the software vendor itself and are accessed by the customer via web technologies. Vendors charge for their software service based on usage dependent metrics on a continuous or monthly basis (Sääksjärvi et al. 2005). The Software-as-a-Service concept enhances the similar service-driven business model of Application

Service Provisioning (ASP) by the use of new technologies and concepts, like the multi-tenancy architecture or service-oriented architecture (SOA) and therefore overcomes the majority of pit-falls of earlier concepts with regard to scalability and reduction of operations cost.

Considering the Software-as-a-Service offerings of companies, it must be noted that nearly every company has a slightly different understanding of Software-as-a-Service. There have been a few approaches to establish a common understanding on the notion “Software-as-a-Service". For instance, Mäkilä et al (2010) investigated the definition of Software-as-a-Service in state-of-the-art literature. They identified five characteristics of Software-as-a-Service (used through a web browser, standardized solution, no installation at customer, no installation work, price based on actual usage) which a company must meet to be considered as “pure Software-as-a-Service" company. These characteristics were empirically analyzed in the Finnish software market. The set of Software-as-a-Service characteristics as established by Mäkilä et al (2010) shows that the concept distinguishes from traditional off-the-shelf software products by more than the aforementioned distribution and pricing model. Further characteristics such as the use of web-technologies and a high degree of software standardization have to be taken into account.

Authors analysing the impact of the Software-as-a-Service concept yield further aspects and characteristics that require attention. Saeed & Jaffar-Ur-Rehman (2005) analyze, from a rather technical perspective, the requirements which are related to a paradigm shift from a product-based model to Software-as-a-Service. The authors consider ultra-late binding of services as the key point of service-based software engineering, highlighting the fine-granularity of solutions. Other impacted aspects are the maintenance of software, service configuration, and how customers are involved in software development and evolution. Building on these results, Olsen (2006) discusses the software engineering practices comparing three aspects (planning, versioning, and maintenance) of product and service-based offerings applying also a management perspective. He concludes that practices need to be adjusted to the new context to compete in the marketplace and to allocate resources appropriately, for instance by shifting the focus from new customers to maintaining long-term stable relationships with the existing customer base. With an organisational perspective, Heart et al. (2010) apply innovation classification theory and the seven fundamental organizational capabilities model in order to determine vendors' readiness to Software-as-a-Service. The authors agree that the concept is revolutionary though various providers in their case study consider it rather evolutionary. The actual impact on the organizations' processes for software delivery is considered marginal. Nonetheless, the data indicates that some capabilities of vendors, especially, sensing the stakeholders, become prevalent. Focusing on the skills of the people delivering the software, Manford (2008) derives a skill catalogue, that takes the changes of the business model into account. After evaluating different opinions and a pro and con discussion of the Software-as-a-Service concept, Manford (2008) is convinced that the concept offers a valid value proposition that is fundamentally different to anything that preceded it and that it requires a different skill set. Especially the shifting responsibility together with the increased complexity of multi-tenant applications is seen as the drive for change. Haines (2007) analyzes the impact again from a more technical perspective and places a special focus on the common Software-as-a-Service enabling technology: web services. The author declares five influences on methodology choice: environment, technology, organization, individual, and task. The analysis and conducted case study indicate required changes in development skills, roles, process, and organizational culture, all pushed by the use of web technologies (Haines 2007). Stuckenberg and Heinzl (2010) investigate the key characteristics of Software-as-a-Service and assess their impact on various existing software and service development methodologies in order to evaluate their applicability for Software-as-a-Service development. The authors conclude that existing methods are not appropriate and require an extension by operations and service related aspects and especially highlight the continuous character of solutions.

Based on the results of the discussed authors, two classes of Software-as-a-Service characteristics can be distinguished (Table 1). On the one hand, there are characteristics which a software offering must fulfil in order to be considered Software-as-a-Service – *defining characteristic*s. On the other hand, there are *supportive characteristics*, which solutions should support in order to be operable or which are derived directly from the defining characteristics.

The two defining characteristics are the central aspects of the concept's distribution and pricing model. The first characteristic is that the solution is *vendor-hosted (VH)*. Key to the Software-as-a-Service concept is the on-demand property, i.e. the customer does not need to host, maintain, and operate any software on his own. Instead the vendor or a third party takes responsibility to host the whole system. Furthermore, the vendor takes responsibility to keep the Software-as-a-Service application up-to-date and to manage incidents and problems. Second, the pricing and licensing model describing characteristic, that the solution is *paid-per-usage (PP)*. Central to the concept is that customers do not buy a software license to use certain software unlimitedly. Instead an organization subscribes to the Software-as-a-Service application to use the software remotely. Based on the subscription, the organization is billed according to the actual usage.

With regard to the supportive characteristics, five aspects can be derived from the literature discussion. (1) *Standardization (SD)*, as Software-as-a-Service solutions should balance the requirements of a broad customer base by providing a core of shared features but also need to address the specific demands of organizations to adapt the system to their own business processes. (2) The use of *web technologies (WT)*, as Software-as-a-Service solutions should be accessible from anywhere without any extensive installation on client-side. Therefore, the UI of Software-as-a-Service solutions is commonly based on open web standards, providing users with the ability to access the application with a standard web browser. From an architectural perspective, the Software-as-a-Service solutions should be open to other applications and provide the necessary flexibility. (3) Use of a *multi-tenancy (MT)* architecture, as Software-as-a-Service application should be designed in a way that an instance of the application can handle multiple customers with their users at once in order to use the resources more efficiently. (4) *Fine granularity (FG)*, as a Software-as-a-Service application should be adapted to changes or to specific customer needs since the default version is standardized. Decomposing the application into fine grained components increases the flexibility and the ability to react to changes and updates. (5) And lastly, the existence of some kind of *continuous evolution (CE)*, since Software-as-a-Service is hosted and operated by the vendor, the vendor is able to continuously evolve the Software-as-a-Service solution instead of introducing updates gradually.

	Defining characteristics		Supportive characteristics				
	VH	PP	SD	WT	MT	FG	CE
Saeed & Jaffar-Ur-Rehman (2005)			X			X	X
Sääksjärvi et al. (2005)		X	X			X	X
Olsen (2006)		X	X		X	X	X
Haines (2007)						X	
Manford (2008)	X	X		X	X	X	
Heart et al. (2010)	X	X		X		X	
Stuckenberg and Heinzl (2010)	X	X	X	X	X	X	X
Mäkilä et al. (2010)	X	X	X	X			

Table 1: Software-as-a-Service characteristics

2.2 Developing Software-as-a-Service

Based on the distilled characteristics, the literature was reviewed regarding research that proposes development and operations methodologies addressing the specific requirements of Software-as-a-Service or a subset of the discussed characteristics. Though methods may be claimed by the respective researchers to be specifically designed for Software-as-a-Service, the underlined definition may differ. For the literature review, the sources were queried in various databases using the term "Software-as-a-Service/SaaS" as a search criterion besides other criteria such as "method", "methodology", and "process". In order to investigate the methods for related characteristics, the term "SaaS" was replaced by the names of the respective characteristics. Furthermore, different synonyms of the terms have been taken into account. Among the results, research articles, concerned particularly with designing development and operation processes but not only single activities, were selected. Table 2 lists the gathered results together with the applied search term category and the characteristics that are

supported by the respective method. Due to space restrictions, the following closer description of methods is limited to those that claim to be adequate for the Software-as-a-Service context.

Espadas et al. (2008) developed a software development method focusing on Software-as-a-Service platforms by comparing the software life cycle of traditional and service-based software. The underlying Software-as-as-Service definition mainly covers the two aspects of multi-tenancy and fine granularity. Following a sequential waterfall-model, Espadas et al. (2008) analyse the different phases of software development and suggest, for instance, that requirements engineering has to be extended by scalability and vendor platform specific requirements and analysis. Due to the platform perspective of the analysis, the suggested implications focus on incorporating platform requirements and the platform's target customers into the different phases. The authors focused and aligned their method on the development of multi-tenancy and fine-granular applications. However, the method does not consider all characteristics. Important aspects such as continuous evolution of software are neglected.

	Search Term	Supported Characteristics						
		VH	PP	SD	WT	MT	FG	CE
McConnell (1996)	CE							X
Coda et al. (1998)	WT				X			
Tomer and Schach (2000)	CE							X
Gaedke and Gräf (2001)	WT				X		X	X
Gnaho (2001)	WT				X			X
Norton (2001)	WT				X			X
Ramesh et al. (2002)	WT				X			X
Dogru and Tanik (2003)	FG						X	
Gronmo et al. (2004)	FG			X			X	
Karhunen et al. (2005)	FG						X	
Crnkovic et al. (2006)	FG						X	
Espadas et al. (2008)	SaaS					X	X	
Singh (2008)	SaaS				X			
Toffolon and Dakhli (2008)	CE							X
Benefield (2009)	SaaS		X					
Guo & Wang (2009)	SaaS	X	X					
La & Kim (2009)	SaaS			X	X	X		
Agarwal (2011)	SaaS							X

Table 2: Review of Software-as-a-Service-related software development methodologies

Singh (2008) discusses the limitations of traditional SCRUM in the context of Software-as-a-Service, as the end-user interaction with the application and the frequent product releases challenge the development processes. Singh (2008) argues that SCRUM is concentrating on project completion rather than enhancing the user experience, and is thus not appropriate for Software-as-a-Service. Based on this limitation, a new SCRUM-based method named U-SCRUM, which takes the limitation in usability into account, is proposed. U-SCRUM incorporates a new role – the usability product owner – being responsible for the user experience, vision definition, along with the user-centred design.

Benefield (2009) analyses lean practices of the manufacturing industry and how they can be applied on software, in order to help the software industry to cope with the arising challenges connected to Software-as-a-Service. Methods like Poka Yoke, Jidoka, Kaizen or Just-in-Time can be used to optimize processes and avoid mistakes. Benefield (2009) argues that the application of these "philosophies" are required due to the new business model, in particular due to the paid-per-usage characteristic.

Guo & Wang (2009) define incident management models for Software-as-a-Service grounded on ITIL that include the customers with their users and the Software-as-a-Service providers with integrators and independent software vendors (ISVs). Although the proposed incident management models for

Software-as-a-Service are rather focused on the customer side, they provide aspects on how IT service management can be implemented and how the customer can be included into the processes.

Another software development process specifically designed for Software-as-a-Service is the model by La & Kim (2009). The authors define an eleven-phase model based on a pre-defined Software-as-a-Service meta model. As a decision basis for their method they define characteristics and desired properties of Software-as-a-Service which have to be taken into account by a software development processes. The model has a strong focus on commonality, i.e. increase reuse of a component, and variability.

Agarwal (2011) describes an agile management method for Software-as-a-Service based on Type C SCRUM. The method is focused on the fast evolution of Software-as-a-Service neglecting other identified characteristics. The method recommends splitting the Scrum team into three sub-teams, each operating in one of the three phases: planning, development, and quality assurance. Agarwal (2011) evaluates the presented method by applying it to a real world project of a web-based transaction processing application.

With regard to research that only addresses a subset of the derived characteristics and does not specifically claim to be designed for the Software-as-a-Service context, it can be noted that the majority of studies only addresses the specific requirements of the characteristic that they have been grouped to. Research in the context of web technologies (WT) additionally tends to take aspect of continuous evolution (CE) into account. However, due to the limited scope of the evaluated studies with regard to the complete list of characteristics, a transfer of the proposed models and use within the Software-as-a-Service context seems not suitable. Though, a final judgement may only be possible after the influence and importance of the different characteristics for this specific context have been analysed.

The literature review reveals that research on Software-as-a-Service development and operations processes is still in an early state. The literature offers first promising suggestions on how to address certain aspects of the Software-as-a-Service concept. However, the analysis and conceptions within the covered studies appear not to take the complete picture of Software-as-a-Service into account. Instead, the applied concept definitions differ from the common understanding of the concept within the general Software-as-a-Service literature and leave a research gap as a methodology addressing the majority of Software-as-a-Service-related aspects is still missing. Also, the proposed methodologies do not provide clear hints on the crucial concept characteristics that drive change requirements. Authors that maintain traditional software engineering perspectives in their evaluation, e.g. by discussing typical phases like requirements engineering, analysis, design, implementation, and testing, tend to focus on addressing rather technical aspects and characteristics. The focus is set on standardization, web-technologies, multi-tenancy or the fine granularity of functions. Authors that apply a more general perspective include the concepts defining characteristics and the continuous evolution into their discussion, but tend to be less specific in their suggestions.

3 RESEARCH METHODOLOGY

In order to address the identified gap and further substantiate change requirements for Software-as-a-Service-related development methodologies, it is necessary to enhance the rather conceptual approaches within the literature with empirical evidence from software vendors actually implementing the Software-as-a-Service concept. As the aim of this research is to provide a first indication in this regard, the following section will introduce the selected research approach.

In order to obtain comparable results within this multi-case research, the site selection focused on companies of a similar context. The selection criteria especially emphasised on the similarity of the produced solutions. Also only solutions with a higher degree of complexity were considered, as only complex software solutions force companies to sufficiently define and utilize development methodologies. Examples of such solutions might be enterprise applications like enterprise resource planning (ERP), customer relationship management (CRM), or business intelligence (BI) solutions. As this study focused on the format and drivers of change, an emphasis was set on companies with

experiences in the traditional as well as the Software-as-a-Service business model. In sum the following criteria had to be fulfilled by the potential sites: (1) its portfolio must comprise on-premise software products and Software-as-a-Service software services. (2) The software to be developed must have a high complexity (i.e., ERP, CRM, or BI solutions). (3) The indirectly compared methodologies for the products and services must be for the same type of application. The data collection comprised mainly semi-structured interviews with different decision takers and people who are directly involved in the development process. The resulting sample for this multiple-case study consists of two companies of large size that have amongst others ERP and CRM solutions in their product portfolio. Both companies have their origin in the on-premises business model and operate in multiple markets worldwide. The Software-as-a-Service solutions analyzed within this case study have been available on a limited number of markets and restricted to selected customer groups for at least four years. As the companies' on-premises solutions may also cover some of the Software-as-a-Service related characteristics, Table 3 captures the results of an evaluation of the solutions with regard to the deducted characteristics of the literature review. It has to be noticed that even the on-premises solutions fulfil the characteristics to a certain extent, or are currently developed in this direction.

Characteristics	Alpha		Beta	
	Service	Product	Service	Product
Vendor-Hosted (VH)	Yes	No	Yes	No
Paid-per-Usage (PP)	Partly	No	Partly	No
Standardization (SD)	Yes	Partly	Yes	Yes
Web-Technologies (WT)	Yes	Partly	Yes	Yes
Multi-Tenancy (MT)	Yes	Partly	Yes	No
Fine-Granularity (FG)	Yes	Partly	Yes	Yes
Continuous Evolution (CE)	Yes	No	Yes	No

Table 3: Software-as-a-Service characteristic fulfilment of offered solutions

The data analysis followed different recommendations of authors in the field of qualitative data analysis (Miles and Hubermann 1994). Based on the transcripts of the interviews, the data was coded in an iterative manner. The complete coding was conducted using QSR Nvivo 9.

4 RESULTS

In the following section, the key observations of the conducted case studies are presented. The presentation is divided into a discussion of the perceived degree of influence of the concept on processes, an analysis of the impacted phases of classical software development and a summary of suggested requirements for development methodologies to make them fit into the Software-as-a-Service context.

4.1 Perceived Degree of Software-as-a-Service Influences

As the analysis of the impact of Software-as-a-Service on development and operations processes in the literature is rather conceptual with little empirical evidence, the initial phase of the conducted case studies briefly queried the perceived degree of Software-as-a-Service driven change of the development methodologies (Table 4). The overall perception supported the claim that an impact can be observed and that it can be traced back to the concept as a whole or to its key characteristics. None of the questioned experts denied an influence.

In case *alpha*, a slightly more critical opinion towards the degree of the influence of Software-as-a-Service on conducted adjustments to the development methodologies can be observed. Changes are perceived to be caused by general trends in the software industry (incl. Software-as-a-Service) and aspects that concern on-premises software to a similar extent. However, the pressure to drive changes

was perceived on a much higher ground in the context of Software-as-a-Service, based for instance on the stronger exposure to the market forces and a resulting greater need for innovation. In these cases, Software-as-a-Service solutions are considered to maintain a pioneering role in further developing the software development methods.

Development processes are influenced by...	Alpha						Beta			Σ
	α1	α2	α3	α4	α5	Σα	β1	β2	Σβ	
... Software-as-a-Service to a large extent.			X	X		2	X	X	2	4
... Software-as-a-Service in selected aspects	X				X	2			0	2
... Software-as-a-Service not at all.						0			0	0
... general industry trends (incl. Software-as-a-Service)		X				1			0	1

Table 4: Perceived degree of Software-as-a-Service influences

Both cases indicate a convergence of introduced features of development processes that address Software-as-a-Service specific aspects, into traditional on-premises product development processes. In general, the influence of trends in the software industry cannot be completely neglected. Therefore, it has to be taken into account that Software-as-a-Service is a major drive for methodological change, but trends may cause a certain bias when analyzing the data.

4.2 Changes in the Software Development Methodology

The following section analyses the Software-as-a-Service impact on different phases of software development and operation based on the IEEE software life cycle. The association with a specific phase is based on explicit statements of the interviewees or implicitly by assigning the outlined changes to a specific phase. Table 5 illustrates the frequency of mentioned aspects by phase. Though a high number does not directly imply the same number of changes, it gives a first but limited impression of phases that are previewed to be impacted to a greater extent. It has to be noted that the reasons why a specific phase is not evaluated as being impacted by a participant may differ and does not directly imply that the phase is not impacted. The expertise of the experts varied, and so were the topics they elaborated in detail.

In case *alpha*, only one participant mentions an impact in the *concept* phase. In-house hosting of Software-as-a-Service solutions provides additional data sources that support portfolio and investment decisions. The *requirement* phase is seen by all experts as being influenced. Aspects mentioned in this context are, for example, the changed customer involvement to determine the priority of a specific requirement, or additional information channels to gather fast feedback on developments. The *design* phase is seen to be impacted by the early consideration of operations-related aspects, like the support of multiple tenants, flexibility and the integration of additional systems. The *implementation* is primarily impacted by similar aspects but also by the implementation of service-oriented principles or database-related aspects. *Testing* is influenced by a parallel development and testing process. The Software-as-a-Service concept has been changing the way *alpha* copes with installations and upgrades during the *installation* and checkout phase. Compared to the traditional shipment, this phase is considerably shortened simply because the solution is vendor-hosted. Another important aspect of the upgrade process is the increasing need for change management since the introduction of a new or changed feature will affect directly a multitude of customers. On the one hand, there are customers who are eager to have the newest functionality. On the other hand, there are those who are satisfied with the current feature set and are reluctant to any change. *Alpha* strives for operating only a limited or even just one version of the software in order to reduce the operating and maintenance costs. Consequently, each customer must be involved in the upgrade and change process. A further aspect which needs to be taken into account is that the system might be business critical for the customer. Hence, any down-time of the system might pose problems for the customer. When a down-time is inevitable, it is necessary to consider that there might be customers on the system from different regions with different working hours. An important aspect to cope with these challenges is the automation of the installation and adaptation process. The actual hand-over process of the code from

the development department to the production and operations department is, however, not further impacted. With regard to the *operation and maintenance* phase, the concept implied responsibility shift is seen to have a big impact. Optimized operations to reduce costs is a major aspect and is realized by operating multiple customers on the same version, the same system, and the same hardware, benefiting of the respective economies of scale. Only one participant referred to a potential change within the *retirement* phase, pointing out the yet unsolved question of the retirement scope ranging from whole services, specific components or single functions.

Software Lifecycle Phase	Alpha						Beta			Σ
	α1	α2	α3	α4	α5	Σα	β1	β2	Σβ	
Concept		2				2		1	1	3
Requirements	2	4	1	4	2	13	3	1	4	17
Design		1	1		1	3	1	3	4	7
Implementation	1	1		1	1	4		1	1	5
Test	5	2	2	2		11		3	3	14
Installation	1		1	4	1	7	3	3	6	13
Maintenance & Operation	1	1	2	8	3	15	3	3	6	21
Retirement	1					1			0	1

Table 5: Impacted phases

In case *beta*, the *concept* phase is considered impacted due to an increased value of the integration of new concepts into the existing technological infrastructure. The *requirements* phase is impacted by the utilization of new information sources for the requirement definition and the feedback channel of the operations team concerning performance tests and benchmarking. The *design* phase is influenced by more frequent software changes. In addition, the design specification needs to take into account a development in smaller increments and integration capabilities with other software systems. *Testing* is changed due to the ceased production environment variation and focusing on one copy of the actual operating environment. Especially, the upgrade procedure and the involvement of the customer play an important role in the *installation* phase. The question whether a customer wants to be upgraded to a higher level or not has to be taken into account when installing a new version. A concise characteristic of the change within the *operation and maintenance* phase is the introduction of a new department which is dedicated to the activities in this phase. Beside the operation and maintenance tasks, this department has a further activity which is the provisioning of feedback to the development department. The final *retirement* phase was not mentioned by participants of case *beta*.

Characteristics	Alpha					Beta		Σ
	α1	α2	α3	α4	α5	β1	β2	
Vendor-Hosted (VH)	X	X	X	X	X	X	X	7
Paid-per-Usage (PP)			X	X	X	X	X	5
Standardization (SD)	X		X					2
Web-Technologies (WT)	X	X						2
Multi-Tenancy (MT)	X					X		2
Fine-Granularity (FG)	X	X					X	3
Continuous Evolution (CE)	X	X	X		X		X	5

Table 6: Characteristics that drives change

Across both cases, it can be noted that not only the obvious *maintenance and operation* phase but also the *requirements* phase is mentioned frequently. In contrast the peripheral phases *concept* and *retirement* seem to be neglected by most of the participants. Stated changes, in particular, address the integration of new information sources into the definition of requirements and the link of operations and development activities. As the expertise of the interviewees varied, so were the scope and detail of corresponding statements concerning the respective phases. In some cases, the statements about

changes in phases, which are outside the participant's scope, were sometimes speculative and could not be taken into account for the analysis.

With regard to the drivers of the identified changes (Table 6), the analysis reveals a pattern that indicates a high perceived impact of the defining characteristic, that solutions are hosted and operated by the vendors and the cost pressure induced by the lower but continuous revenue stream of the pay-per-usage pricing scheme. From the group of supportive characteristics, only the pressure to continuously evolve the solution was mentioned frequently throughout the interviews. The other characteristics were only mentioned when the experts dived deep into a technical level of software development.

4.3 Requirements of Software-as-a-Service Development and Operations Process Designs

Based on the identified implications of the different phases of software development and operation, the following section presents identified process requirements that shall be taken into account when a development methodology is adapted to or applied in the Software-as-a-Service context (Table 7). Due to the continuous improvement of development approaches within the companies, the participants were also asked about future modification plans. The expert statements are therefore not necessarily based on existing processes within their companies but also refer to current problems and process limitations and approaches to tackle them.

	Alpha						Beta			Σ
	α1	α2	α3	α4	α5	Σα	β1	β2	Σβ	
Analytical Customer Integration	X	X	X	X	X	5	X	X	2	7
Increased Automation		X	X	X	X	4		X	1	5
Integration of Operations		X		X	X	3	X	X	2	5
Avoidance of Waste	X	X	X	X		4			0	4
Increased Flexibility	X	X	X			3		X	1	4
Increased Governance	X		X		X	3		X	1	4
Shorter Release-Cycle	X	X			X	3		X	1	4
Increased Security Awareness	X					1	X		1	2
Continuous Cost Controlling				X		1			0	1
Role: SaaS Designer					X	1			0	1
User Centric Development	X					1			0	1

Table 7: Identified process requirements

In both cases, frequently mentioned aspects are the approaches for how the customers and users are integrated into the software development process. These aspects mainly concern changes in the *requirements* and the *operations and maintenance* phase, that address the Software-as-a-Service concept's altered deployment model and resulting direct customer relationship of the software vendor. The experts of case *alpha* in particular mentioned required process adjustments to take the given opportunities to evaluate actual system usage to feed analytical requirements engineering, test requirements in small scale experiments that may be embedded in the live systems, or collect feedback derived from integrated direct communication channels with end-users, like forums or chats. As a result of the changed target group of primarily small and medium-sized enterprises, the customer base consists of many smaller companies instead of a small number of key customers. Therefore, the customer integration is oriented at (anonymous) analytical approaches based on the whole customer base, rather than customer interactions at a personal level. In case *beta*, the experts described established integration structures between development and operations to safeguard the quality and operability of solutions.

Addressing changes within the *testing* and *operations and maintenance* phase, experts mention requirements to increasingly automate processes and activities. Due to the shift of operations, economies of scale gain importance, requiring testing, installation, and maintenance activities to be automated and thus reduce the time of the involved procedures. Similarly, methods and principles

need to be applied to avoid waste, for example in form of inefficient algorithms, unused functions and unnecessary complexity. Experiences from model-driven development may be of advantage.

The experts also see an increased requirement of flexibility and agility within the development. Though agile development and management methods can be found in on-premises development as well, the need for such methods in the service context is much stronger. Perceived reasons for this are the fixed and shortened release cycles that only allow a variation of scope and not time and quality. Additionally, the development organizations have to be able to react to fast changing requirements in the market as well as the increase innovation expectation of Software-as-a-Service customers, without being captured in rigorous processes. The interviews also revealed an increased requirement for a governance layer throughout the entire development and operations processes, which safeguards the interference and distraction of the customers' business operation. As all changes to the software system immediately affect all users, the impact on the customer's daily operations has to be minimized. From a technical perspective, dependencies have to be carefully managed to cope with the complexity and the fast pace of Software-as-a-Service evolution. Governance efforts may decrease setups and unwished work-arounds and reduce the number of concurrent versions to a minimum of one, as the optimum of a multi-tenancy architecture. In case *alpha*, this optimum was however not reached, as migration of the productive system could not be realized at once, leaving the company to run up to three software versions at the same time.

5 DISCUSSION

Reflecting the studied cases with the literature yields varying relevance of certain Software-as-a-Service characteristics with regard to their influence on development processes and activities. Proposed Software-as-a-Service development processes of the literature often focus on technical aspects that are reflected by the supportive characteristics. The cases, however, indicate that practitioners perceive the biggest impact in aspects like the solutions being hosted by the vendor.

To some degree, this can be explained by the comparison in Table 3, as certain characteristics are not Software-as-a-Service specific but also found in traditional on-premises solutions. Standardization for instance is a general challenge of companies offering applications to a broad customer base by trying to aggregate and thus standardise functionality that fulfils the requirements of diverse groups of customers (Brehm et al. 2001). In the Software-as-a-Service context, this demand motivated perspective to standardize is extended with a supply oriented perspective, as standardization and the provisioning of identical solutions to all customers does not only save development resources per sold solution but also eases the operation of the solution and reduces the occurring costs. This increases the relevance of standardization aspects within the development, but does not render it unique for Software-as-a-Service. A similar argument can be made for the fine-granularity that among other objectives allows balancing between standardization and offering solutions to differing customer requirements. The third characteristic that appears already common in the product business is the use of web technologies. Due to client-server architectures being common already with on-premises application, web technology related issues are also of similar relevance in the traditional business.

As a result, some of the identified process requirements are also not unique to Software-as-a-Service and are already addressed by existing development methodologies. A core value of agile development, for instance, is dealing with the avoidance of waste and aiming on increasing flexibility and shortening release-cycles. Other requirements, like the integration of operations, seem less elaborated in existing development methods.

Overall, the combination of the defining characteristics results in an increased cost as well as innovation pressure. On the one hand, the vendors' responsibility is increased as additional activities are added, but at the same time the revenue stream is reduced or delayed. This situation pushes vendors to optimize and streamline existing practices to reduce internal costs. Increased automation of processes and the reduction of unnecessary software functions or handling activities are examples in this area. On the other hand, the business model is pushing vendors to innovate quicker and to continuously convince the customer of their solutions. Customers have increased expectations on

receiving more frequent updates and prompt responses to arising problems. Vendors are analysing the potential to take advantage of the direct customer interaction and use the resulting information to actively adapt solutions to fit the customers' expectations. The combination of these aspects asks for an increased integration of internal activities like development and operations, as well as an increased customer-orientation and integration.

The findings of this study provide details about the impact of the Software-as-a-Service concept on development processes of software vendors. A clear influence of the concept is identified in the literature as well as in the conducted case studies. The findings identify the driving forces behind the concept and provide current approaches and challenges of Software-as-a-Service vendors. Identifying and evaluating the key characteristics of the Software-as-a-Service concept helps to substantiate the discussion around the concept and to structure future research. The findings are also of value for companies new to the Software-as-a-Service concept or currently planning to offer their solutions as Software-as-a-Service in the future. The results demonstrate that offering Software-as-a-Service is far more complex than just changing the distribution and pricing model of existing solutions and requiring thorough planning and adjustments to processes and practises.

The study has a few limitations that will be addressed in future research. The case sample can only provide a small and limited picture with regard to the raised research question. Further companies need to be included to be able to draw a more complete picture of the influence of Software-as-a-Service. Pure Software-as-a-Service vendors that were excluded from the sample in this study may also provide additional insights. As this study is part of a bigger research project, these limitations are currently addressed. Furthermore, the presented literature is limited to research articles that analyse the complete software development lifecycle, neglecting articles that focus on specific phases, like requirements engineering. A more detailed analysis of single phases may provide further support to the research question. Lastly, this study takes a rather traditional perspective on software development, using the IEEE software lifecycle to structure the analysis. This approach may have limited the perspective on service characteristics, as lifecycle models in the field of service development often follow a different structure.

6 CONCLUSION

The objective of this study was to analyse the influence of the Software-as-a-Service concept on software development processes and to identify and discuss the driving forces behind the concept. For this reason, the literature was reviewed to derive the key characteristics of the Software-as-a-Service concept and two defining characteristics (vendor-hosted, pay-per-usage) and five supportive characteristics (standardization, web-technologies, multi-tenancy, fine-granularity, continuous evolution) were identified. The subsequent evaluation of existing methodologies disclosed, that all approaches only address a subset of the identified characteristics.

The experts in the conducted case studies of vendors that offer traditional on-premises software as well as Software-as-a-Service perceive the concept to impact development processes in certain aspects or to a large extent. The analysis revealed further insights into the impacted phases, as well as the driving forces behind potential change. Activities that in the traditional view fall into the phases of requirements engineering and operations are perceived to be impacted the most. The required changes are associated with the defining characteristics of the concept as well as its continuous evolution character. Vendors have or are currently undergoing a redesign of their processes to be able to facilitate the potential of analytical customer integration, as well as an increased integration of different internal activities like development and operation. Two aspects that still adhere a variety of problems and challenges and that seem worth being looked at in more detail in future studies.

References

Agarwal, P. 2011. "Continuous Scrum: Agile Management of Saas Products," *4th India Software Engineering Conference*, Kerala, India: ACM, pp. 51-60.

Benefield, R. 2009. "Agile Deployment: Lean Service Management and Deployment Strategies for the Saas Enterprise," *42nd Hawaii International Conference on System Sciences*, R.H. Sprague (ed.), Waikoloa, Hawaii: IEEE.

Brehm, L., Heinzl, A., and Markus, L. 2001. "Tailoring Erp Systems: A Spectrum of Choices and Their Implications," *34. Hawaii International Conference of System Science*, Maui, Hawaii, USA.

Coda, F., Ghezzi, C., Vigna, G., and Garazotto, F. 1998. "Towards a Software Engineering Approach to Web Site Development," *The Ninth International Workshop on Software Specification and Design*: IEEE, pp. 8-17.

Crnkovic, I., Chaudron, M., and Larsson, S. 2006. "Component-Based Development Process and Component Lifecycle," *International Conference on Software Engineering Advances*: IEEE, pp. 8-17.

Dignan, L. 2008. "Sap's Apotheker: Business Bydesign Costs Led to Delay." from http://www.zdnet.com/blog/btl/saps-apotheker-business-bydesign-costs-led-to-delay/8724

Dogru, A., and Tanik, M. 2003. "A Process Model for Component-Oriented Software Engineering," *IEEE Software* (20:2), pp. 20-27.

Espadas, J., Concha, D., and Arturo, M. 2008. "Application Development over Software-as-a-Service Platforms," *The Third International Conference on Software Engineering Advances*, H. Mannaert, T. Ohta, C. Dini and R. Pellerin (eds.), Sliema, Malta, pp. 97-104.

Gaedke, M., and Gräf, G. 2001. "Development and Evolution of Web-Applications Using the Webcomposition Precess Model," in *Web Engineering 2000*, M. S. and D. Y. (eds.). Berlin, Heidelber, Germany: Springer-Verlag, pp. 58-76.

Gartner. 2009. *Fact Checking: The Five Most-Common Saas Assumptions* Gartner.

Gnaho, C. (ed.) 2001. *Web-Based Infromation Systems Development - a User Centred Engineering Approach*. Berlin, Heidelberg, Germany: Springer-Verlag.

Gronmo, R., Skogan, D., Solheim, I., and Oldevik, J. 2004. "Model-Driven Web Services Development," *International Conference on Research Challenges in Computer Science*: IEEE, pp. 42-45.

Guo, W., and Wang, Y. 2009. "An Incident Management Model for Saas Application in the It Organization," *International Conference on Research Challenges in Computer Science*: IEEE.

Haines, M.N. 2007. "The Impact of Service-Oriented Application Development on Software Development Methodology," *40th Annual Hawaii International Conference on System Sciences*.

Heart, T., Tsur, N.S., and Pliskin, N. 2010. "Software-as-a-Service Vendors: Are They Ready to Successfully Deliver?," in *Global Sourcing of Information Technology and Business Processes*, I. Oshri and J. Kotlarsky (eds.). Berlin, Heidelberg: Springer pp. 151-184.

Juell-Skielse, G., and Enquist, H. 2012. "Implications of Erp-as-Service: Re-Conceptualizing Enterprise Information Systems," C. Møller and S. Chaudhry (eds.). Berlin, Heidelberg: Springer pp. 129-151.

Karhunen, H., Jantti, M., and Eerola, A. 2005. "Service-Oriented Software Engineering (Sose) Framework," *International Conference on Services Systems and Services Management*, pp. 1199-1204.

La, H., and Kim, S. 2009. "A Systematic Process for Developing High Quality Saas Cloud Services," in *Cloud Computing*, M. Jaatun, G. Zhao and C. Rong (eds.). Berlin, Heidelberg: Springer pp. 278-289.

Mäkilä, T., Järvi, A., Rönkkö, M., and Nissilä, J. 2010. "How to Define Software-as-a-Service – an Empirical Study of Finnish Saas Providers," in *Software Business*, P. Tyrväinen, S. Jansen and M.A. Cusumano (eds.). Berlin, Heidelberg: Springer, pp. 115-124.

Manford, C. 2008. "The Impact of Saas Model of Software Delivery," *21st Annual Conference of the Natgional Advisory Committee on Computing Qualifications*, Auckland, New Zealand, pp. 283-286.

McConnell, S. 1996. *Rapid Development: Taming Wild Software Schedules*. Redmond, WA, USA: Microsoft Press.

Miles, M.B., and Hubermann, A.M. 1994. *Qualitiative Data Analysis: An Expanded Sourcebook*. Thousand Oaks, CA: Sage Publications.

Norton, S. (ed.) 2001. *Applying Cross-Functional Evolutionary Methodologies to Web Development*. Berlin, Heidelberg, Germany: Springer-Verlag.

Olsen, R. 2006. "Transitioning to Software as a Service: Realigning Software Engineering Practices with the New Business Model," *IEEE International Conference on Service Operations and Logistics, and Informatics*, Shanghai, China: IEEE, pp. 266-271.

Ramesh, B., Pries-Heje, J., and Baskerville, R. 2002. "Internet Software Engineering: A Different Class of Processes," *Annals of Software Engineering* (14:1), p. 169195.

Sääksjärvi, M., Lassila, A., and Nordström, H. 2005. "Evaluating the Software as a Service Business Model: From Cpu Time-Sharing to Online Innovation Sharing," *IADIS International Conference e-Society*, P. Isaisas, P. Kommers and M. Mc-Pherson (eds.), Qawra, Malta, pp. 177-186.

Sääksjärvi, M., Lassila, A. and Nordström, H. 2005. "Evaluating the Software as a Service Business Model: From Cpu Time-Sharing to Online Innovation Sharing," *IADIS International Conference e-Society 2005*, P.K.a.M.M.-P. Petro Isaisas (ed.), Qawra, Malta, pp. 177-186.

Singh, M. 2008. "U-Scrum: An Agile Methodology for Promoting Usability," *Agile 2008*: IEEE, pp. 555-560.

Stuckenberg, S., Fielt, E., and Loser, T. 2011. "The Impact of Software-as-a-Service on Business Models of Leading Software Vendors: Experiences from Three Explorative Case Studies," *Pacific Asia Conference on Information Systems*, Brisbane, Australia.

Stuckenberg, S., and Heinzl, A. 2010. "The Impact of the Software-as-a-Service Concept on the Underlying Software and Service Development Processes," *14th Pacific Asia Conference on Information Systems*, Taipei, Taiwan, pp. 1297-1308.

Toffolon, C., and Dakhli, S. 2008. "An Interative Meta-Lifecycle for Software Development, Evolution and Maintenance," *Third International Conference on Software Engineering Advances*: IEEE, pp. 284-289.

Tomer, A., and Schach, S. 2000. "The Evolution Tree: A Maintenance-Oriented Software Development Model," *The Forth European Software Maintenance and Reengineering Conference*, pp. 209-214.

Weinhardt, C., Anandasivam, A., Blau, B., Borissov, N., Meinl, T., Michalk, W., and Stößer, J. 2009. "Cloud Computing – a Classification, Business Models, and Research Directions," *Business & Information Systems Engineering* (1:5), pp. 391-399.

West, M., McNee, B., Geisehecker, L., Guptill, B., McNeill, R., and Burns, C. 2010. "Key Saas, Paas and Iaas Trends through 2015: Business Transformation Via the Cloud." Saugatuck Technology Inc.

Xin, M., and Levina, N. 2008. "Software-as-a Service Model: Elaborating Client-Side Adoption Factors," *29th International Conference on Information Systems*, R. Boland, M. Limayem and B. Pentland (eds.), Paris, France.