1-1-2010

The Impact of the Software-as-a-Service Concept on the Underlying Software and Service Development Processes

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

Armin Heinzl
University of Mannheim, heinzl@uni-mannheim.de

THE IMPACT OF THE SOFTWARE-AS-A-SERVICE CONCEPT ON THE UNDERLYING SOFTWARE AND SERVICE DEVELOPMENT PROCESSES

Sebastian Stuckenberg, Chair of Business Administration and Information Systems, University of Mannheim, Mannheim, Germany, stuckenberg@uni-mannheim.de

Armin Heinzl, Chair of Business Administration and Information Systems, University of Mannheim, Mannheim, Germany, heinzl@uni-mannheim.de

Abstract

Research on Software-as-a-Service has received an increasing attention. However, most of the existing literature focuses on the general business model, its adoption and diffusion or on the technological implementation. The impact potentials on a process level and especially on development processes have rarely been analyzed. The objective of this paper is the impact analysis of the key characteristics of the Software-as-a-Service concept on development processes of Software-as-a-Service vendors. For this purpose, Software-as-a-Service definitions in research publications will be analyzed to extract the key characteristics of the concept. These are subsequently evaluated against their development process impact. Implications for the software and service development processes are derived to form a basis for an assessment of current methodologies. The results suggest the need for a continious process of software development and service operations actvities with a close linkage between the functions and high customer involvment. The service character challanges the development-focused methodologies and the Software-as-a-Service marketing concepts push for a continious flow of enhancements. The centrality of the vendor controlled infrastructure advances the information quality of development decisions by offering possibilities to evaluate user behavior within the system. The current methodologies are, however, not able to cover the new requirements and opportunities.

Keywords: Software-as-a-Service, Software Engineering, Service Engineering, Service Science, Process Impact.

INTRODUCTION

When looking at the Software-as-a-Service concept, a shift of process responsibilities between the customers and software developing companies can be observed. The Software-as-a-Service definition of Sääksjärvi et al. (2005) that describes the concept as applications with location and time independent remote access, based on a multi-tenant infrastructure and new payment models, implies various changes in the structure and sequence of development activities that are yet unconsidered within software development processes. The most obvious change is the missing deployment on the customer side that is replaced by a centralized infrastructure within the responsibility of the software vendor. The traditional understanding of software development processes as advocated by Sommerville (2007) or others with its generic phases "Analyze", "Design", "Implement", "Validate" and "Evolution" addresses the creation of the software artifact, but it offers only limited explanation regarding its use. Despite the growing number of software vendors offering their solutions using a service model, insufficient research effort is put into the validation of the processes and techniques for the developing of software with a service model. The question arises, whether the previously used methodologies uncoupled from the discussion between traditional and agile approaches of software development are still feasible in the new context.

Apart from pure responsibility shifts in the processes, the Software-as-a-Service concept also offers new capabilities, like the possibility to monitor user behavior within the software systems that can be facilitated to improve the development processes, the software itself and in particular the processes implemented within the software. Only a couple of researchers address these new capabilities and their impacts on processes. Choudhary (2007) focuses on the Software-as-a-Service license model and shows that investments in software development and the software quality will increase with the new model. Olsen (2006) compares the product and service models regarding changes in the planning, versioning and maintenance of software. His findings indicate the different roles of the marketing department and the decreased importance of the sales cycle combined with the growing focus on long-term customer relationships to be major causes for the demand to change development processes. His results extend the research done by Brereton et al (1999) which aimed on defining a research agenda. The embedded concept of late-binding (Bennett et al. 2000) is however absent in current Software-as-a-Service offerings like Salesforce.com or SAP Business ByDesign and the predicted consequences are therefore not completely applicable. Espadas et al. (2008) conduct a development phase oriented impact analysis of the Software-as-a-Service concept with focus on development projects that are deployed on third-party infrastructure platform offerings. Their results suggest new constraints and activities to be taken care of in the different phases, like the enhanced requirements of storage and performance scalability and the implications of predefined platform architectures on the processes.

The second yet unsolved question is the influence the service nature has on the development processes. Services like software lack a physical representation and are considered immaterial with similar consequences for the development processes. Though different models for their development exist, the efficiency and validity of these cannot be measured as clearly as a development process for a physical good. The Software-as-a-Service concept combines the software with the service domain and bears the inherent challenges of both domains. It is therefore required to analyze both fields regarding their methods and best practises in order to respond to this increased complexity. A preliminary step is the evaluation of the characteristics of the Software-as-a-Service concept itself and their impacts on the software development processes and in particular the impact of the service character. This knowledge allows in a subsequent step, the evaluation of whether and how activities of the service development can be integrated in the software development process and how insights of the service development domain, with its roots mainly based in the field of marketing, can contribute to form a joined development process for Software-as-a-Service solutions.

1 ORGANIZATION OF THE PAPER

To analyze the potential effects of the Software-as-a-Service concept on the development processes of software companies, the following research questions will be addressed:

- Which Software-as-a-Service characteristics challenge the current practice of software development?
- Are service characteristics important and to what extent are service characteristics challenging the existing software development processes?
- What are the implications for the development process of Software-as-a-Service solutions and to what extend are existing methodologies fulfilling these requirements?

This conceptual paper focuses within the context of discovery on comparing the new Software-as-a-Service domain with the existing software and service domains and identifying interrelations of the three. It therefore applies the theory development strategies 'use analogy' and 'use interrelations' (Darden 1991; Yadav 2010). Based on the existing literature, characteristics of the Software-as-a-Service concept that may have an influence on processes, as well as possible process requirements, will be deducted. The analysis is divided into three parts. At first the Software-as-a-Service concept itself is reviewed regarding its key characteristics and their development process disrupting potential. Process requirements are derived and current development processes from the literature evaluated against these. The collection of requirements is supplemented with further findings of an analysis of the service development literature and practice. The results are used to formulate implications and new opportunities for software companies with Software-as-a-Service offerings and to lay down the foundation for a research agenda to form an integrated Software and Service approach.

The paper is organized as follows. Section 3 comprises a collection of common Software-as-a-Service characteristics from the current literature and an evaluation against their possible process impact. Implications for development processes are outlined before well cited development methodologies are evaluated in section 4. Section 5 summarizes the findings and outlines the contribution to the field.

2 SOFTWARE-AS-A-SERVICE CHARACTERISTICS

To be able to analyze a potential influence of Software-as-a-Service on development processes, it is required to outline the Software-as-a-Service inherent features and evaluate their process determining characters. Therefore table 1 lists the most common Software-as-a-Service characteristics ([C-1] – [C-9]). These are taken from definitions in Software-as-a-Service research papers. The derived implications ([I-1] – [I-9]) are summarized later in table 2.

It is required to separate between typical functionality and process characteristics. A functionality is only further examined when a clear dependency between the function and a specific process requirement is visible, as indicated by the 'impact on development processes' column in table 1. If not stated otherwise, 'process' always refers to the development process as the sequence of activities required to develop software. The process impact is considered 'high' when the characteristic directly influences the process design, 'medium' when the characteristic's impact is focusing on the information quality of the processes or software characteristics which implementation may affect the processes, therefore representing optional and specific implementation dependent characteristics. The impact is regarded as 'low' when the characteristic addresses software attributes or the general business model with limited influence on the processes.

Delivery. For Software-as-a-Service solutions, the vendor does not ship the software to the customer anymore. Instead, it is installed, executed and maintained by the vendor (Sun et al. 2007). This has the following implications for the processes:

- **[C-1] Process extension**. [I-1] The Software-as-a-Service provider needs to extend its processes by activities that take care of operating the software. These include the installation or configuration of the software for the customer, the operation of the required systems, the maintenance and upgrading of the systems' infrastructure and software components, as well as the

enhancement. Whereas in the on-premises software model the deployment and maintenance is of lower importance for the vendor and often done by third party integrators, the service component results in a responsibility shift that makes such activities mission critical for the provider. The typical service attribute of value creation and fulfillment by consumption (also known as the uno-actu principle, which will be discussed as part of C-4) is of high relevance in this context. Update and maintenance activities directly affect the quality perception of the customer. Due to the central (multi-tenant) infrastructure, errors occur during these activities concern all customers simultaneously (Olsen 2006). As a result, the processes to conduct changes to the software are required to be precisely planned and connected with development activities. Combined with the process extension is an additional software functionality prioritizing function that emphasizes on functionality to ease software operation and maintenance and therefore decreases the total cost of ownership for the vendor (Aulbach et al. 2008, Choudhary 2007).

Characteristic	Description	Source (exemplary)	Impact on development processes
[C-1]: Delivery: process extension	Maintenance, operation of software in the responsibility of the vendor.	Buxmann et al. (2008), Sun et al. (2007), Xin et al. (2008)	High
[C-2]: Delivery: continuous enhancement	Version free software without releases and continuous enhancement.	Choudhary (2007), Olsen (2006), Sääksjärvi et al. (2005)	High
[C-3]: Delivery/Software: central infrastructure	The vendor provides the (mainly multi-tenant) infrastructure, one-to-many distribution model.	Aulbach et al. (2008), Buxmann et al. (2008), Sääksjärvi et al. (2005), Xin et al. (2008)	Medium
[C-4]: Service character	The provision of the solution is considered as a service with the typical service attributes.	Buxmann et al. (2008), Lassila (2006), Saeed et al. (2005), Turner et al. (2003)	High
[C-5]: Software: limited customizing	The multi-tenant infrastructure reduces the possibilities to adjust / customize the software to the user's specific needs.	Lassila (2006), Xin et al. (2008)	Medium
[C-6]: Software: SOA	A modular structure of the solution.	Brereton et al. (1999), Olsen (2006), Sun et al. (2007), Turner et al. (2003)	Medium
[C-7]: Software: remote Access	The software is accessed over the internet using a web browser.	Sääksjärvi et al. (2005), Sun et al. (2007), Susarla et al. (2009)	Low
[C-8]: Business model: payment model	Subscription price model, usage-based pricing, no upfront investments.	Choudhary (2007), Cusumano (2008), Susarla et al. (2009)	Low
[C-9]: Business model: ownership	Separation of possession and ownership of software from its use.	Sun et al. (2007), Turner et al. (2003)	Low

Table 1: Software-as-a-Service Characteristics

- **[C-2] Continuous enhancement**. Vendors promote that they provide the user with a solution that represents the latest and most advanced version of development. This marketing strategy results in the expectation of the customer to be supplied with a continuous flow of software enhancements as a prerequisite to prolong the service subscription (Choudhary 2007). Despite the fact that being up-to-date is a software attribute, it has direct implications for the processes since it changes the general conditions of having release based development cycles as well as the need of software versioning. The implications are a need for a modular software structure that allows exchanging single components (compare C-6), [I-2] as well as a development process that facilitates continuous development (Olsen 2006). Baskerville et al. (2003) refer to this as

'internet-speed development' and list parallel release development, dynamic completion-status-based feature scheduling and a lower emphasis on documentation due to the short life-time of software components as process implications. Methods known from agile development are seen as promising (Baskerville et al. 2003).
Software-as-a-Service solutions therefore not only have an impact on the update task and its responsibilities as indicated in I-1, but also on the supply-side by challenging the processes that generate the updates and enhancements.

- **[C-3] Central infrastructure**. The third characteristic is a consequence of the central structure of Software-as-a-Service solutions and to a lesser extent a direct requirement but a chance for process and software enhancements by improving the information quality that is used during the development process. Since the vendor is managing the infrastructure, he has direct access to user interactions with his service similar to website owners. The involvement of the external factor is a major challenge of services that can be leveraged to an advantage in the case of Software-as-a-Service. Possible scenarios include the evaluation of expected and actual user behavior in the software and the direct feedback of frequently used functions. The gathered data can be facilitated to identify functions that require improvements or user interface designs and processes that increase the user's productivity. Software-as-a-Service allows an extended study of the quality of use of the offered solutions. Though these techniques already exist especially in related setups, e.g. for the internet page development (Herder 2009), the objectives are slightly different. Whereas for internet pages, the methods are marketing driven and aim on the search for the most suitable placement of information and advertisements, the objective in Software-as-a-Service solutions is expected to be more process driven and aim on finding the best composition to complete a certain task with the software solution.
  The utilization of web technologies to deliver the solution has resulted in the possibility to seamlessly integrate communities, forums, documentations and learning tools into the offerings. These enhancements not only act as self-service applications with the intention to reduce consulting and training efforts, but also offer another information source for the requirements engineering phase. The interaction of the user with these tools can expose weaknesses of functionality and user interfaces, since the user is more likely to use such tools when concrete problems with the software arise. The tools in addition can be used as a collaboration platform to intensify the interaction between users and developers.
  Though these approaches are mainly focusing on improving the information quality utilized in the development processes, they still require certain process adjustments to be exercised completely. [I-3] At first, the requirements engineering phase need to be extended by activities that gather and take advantage of the new information sources, [I-4] therefore the development functions need to be closer linked with operations functions.
  The possibility to benefit from the new information sources of direct user feedback and the central infrastructure also challenge the test and validation processes. As already outlined in the context of the necessary process extension, errors in the update process do affect all customers at the same time. [I-5] The process therefore needs to be well structured to reduce errors and also should minimize the disruption and disarrangement at the customers' side (Lindholm 2007). The new additional information sources can be used to implement short feedback cycles in the process.

[C-4] Service character. The service-related implications can be divided into those that are a result of the service model, and thus mainly a consequence of the Software-as-a-Service property that the software is managed and executed by the vendor ([C-1] – [C-3]) and those that may be derived from the traditional service research and its suggested development activities. As a prerequisite to consider findings of the latter, it is necessary to analyze the ratio of service characteristics within the Software-as-a-Service concept. Therefore, it is important to understand the defining criteria of services.

Services are understood as independent, saleable activities, which are combined with the supply and/or use of a given capacity. The goal is the combination of internal and external factors to produce a benefit (Heinrich et. al. 2004). Constitutive criteria are therefore the immateriality, the uno-actu-principle and the existence of an external factor within the fulfilment phase. In the context of IT-services the characteristics are extended with the need to facilitate information and communication technologies to support the value creation (Meyer et al. 2008). The degree of IT involvement

separates IT-services in services that are supported by IT, services that go along with IT and combined hybrid solutions (Böttcher et al. 2004). The Software-as-a-Service concept complies with these defining criteria. Software is per definition regarded as immaterial (Heinrich et al. 2004), manifested in the limited perceptual capacity of the quality prior to consumption (Young 2008). Free trial periods try to reduce the uncertainty but cannot guarantee a lasting satisfying quality level during the long-term usage of the solution. At the same time, the promised service capability is dependent on the network reliability and the inherent failure risk. Although software in its source code form can be stored, the capacity required to deliver the value cannot. Excess capacity is wasted, together with the resources used to develop it. The value of the Software-as-a-Service solution is hence achieved through the use of the solution, which is also known as the uno-actu-principle. The external factor in the case of Software-as-a-Service is the user of the solution who needs the ability to use the software according to his needs in order to benefit from the service. Under these considerations, it is valid and promising to analyze service-related development activities in the process of developing a Software-as-a-Service solution.

For the development processes the service characteristics result in an increased emphasis on activities of the delivery and fulfilment phases, which is caused by the concentration of the value creation within these phases. This differs from products which value creation mainly yields from the design and production phases and less from the sales phase. In the service context the development and the delivery cannot be separated that clearly (Ramaswamy 1994). Therefore it can be expected that the development and deployment processes of Software-as-a-Service solutions need to be highly integrated. This finding is very obvious since it is also required by the responsibility shift of deployment related activities to the software vendors which enables the closer integration and lead to implication I-1.

In addition, software and service methodologies underlie a different understanding of quality. Software-as-a-Service therefore asks for a tight connection of activities of both domains. In software development the objectives of quality checks are (1) the reduction of errors that may occur during the execution of the software and affect the quality perception of the customer, (2) the validation of the implementation with the previously defined software requirements. The focus however is very often on the second (Sommerville 2007). Service engineering models on the other hand concentrate on the validation of the requirements with the customers' requirements. This is caused by the service characteristics (Uno-actu-principle as well as the involvement of an external factor) and the direct perception of the customer satisfaction (Ramaswamy 1994). Though software development models (especially agile methods (Beck 2000)) also place priority on customer integration in the development process, the contact is not as intence after the software has been shipped. In a Software-as-a-Service context, however, the customer has to be continuously convinced of the value of the service (Choudhary 2007), which requires both best practices to be linked together to guarantee quality in the development as well as during the execution of the solution. [I-6] The test and validation phases are therefore not single activities that are executed with every change of the software, but are continuously applied and they take the user satisfaction with the service into account. The activities are hence extended by the measurement of the execution quality. The research results of Choudhary (2007) which show that vendors are more likely to invest in quality under a subscription licensing model than under an on-premises licensing scheme, can be seen as a clear indication for this.

Services are in general very hard to standardize. The heterogeneity in form of adjustments that satisfy the individual requirements of the customer is often mentioned as key characteristic (Young 2008). Directly linked to this attribute of service are the characteristics 'Limited customizing' and 'SOA'. The former is a result of the multi-tenant architecture and the necessity of the service providers to reduce their operations costs, while the latter is an often anticipated technology to strike a balance between standardization and customizability. It is therefore debatable if 'SOA' and 'Limited customizing' can be counted as typical Software-as-a-Service characteristic or as a result of the common standardization problem (Brehm et al. 2001) which is further challenged by the uno-actu-principle of service and the embedded need to fulfil specific customer demands. [I-7] An adequate integration of the customer into the processes hence seems inevitable.

Software. This category subsumes Software-as-a-Service characteristics that represent software properties. Though they are often cited in combination with Software-as-a-Service, it has to be questioned whether they are a result of the concept itself or contingent upon the available technological possibilities and therefore represent current implementations and methods to counter the challenges of Software-as-a-Service.

[C-5] Limited customizing. Current Software-as-a-Service solutions offer compared to on-premises solutions only limited possibilities to customize the software to the specific needs of the customer (Xin et al. 2008). This can be attributed to the standardization and operations cost reduction efforts of the vendor and the often chosen multi-tenant architecture (Aulbach et al. 2008). Hence, it can be regarded as a limitation and the Software-as-a-Service-driven process adjustment should therefore focus on resolving the restrictions. Sun et al. (2008) suggest a framework that aims on helping software vendors to take the appropriate steps to deliver a Software-as-a-Service solution that is standardized on the one hand but also includes customizing functionality to address specific needs. The framework emphasizes on activities that screen the market for potential individual needs in order to define customizing options in the early stage of the development. This however can only loosen the limitation. A closer integration of the customer as requested in I-7 is certainly required.

- **[C-6] SOA**. SOA is mentioned by Brereton et al. (1999) and Turner et al. (2003) as Software-as-a-Service characteristic. Brereton et al. (1999) understand Software-as-a-Service yet as software that consists of services that are bind together dynamically as needed. It, thus, offers vast customizing possibilities and the association to characteristic C-5, as one motivation to introduce a SOA, is clearly visible. Their understanding of the Software-as-a-Service concept is however not seen in current solutions which use SOA mainly to integrate with other products (Zencke et al. 2008). The concepts of SOA and Software-as-a-Service have on a high-level view a few things in common. They both advocate the idea of services, yet with a different granularity. Whereas in SOA the service comprise out of a function with standardized interfaces (Erl 2008), is the service in Software-as-a-Service an aggregation of functionality in form of a complete software package that in addition is associated with a business model setup. The existing research results that investigate the implications of SOA on the processes, are therefore also of interest in the Software-as-a-Service context. [I-9] In general can be expected that the processes need to handle modular software components.
- **[C-7] Remote access**. In Software-as-a-Service, the solution is accessed by the customer remotely over the Internet (Sääksjärvi et al. 2005). The process implications of this are however considered low as it is in first place affecting the software architecture and does not differ from remotely accessed client-server software in an on-premises model. The by the Software-as-a-Service concept implied consequence that the systems are managed by the vendor is already discussed in characteristic C-1.

Business model. The often cited Software-as-a-Service specifics of the revenue model [C-8] and the ownership [C-9] as outlined by Choudhary (2007) or Turner et al. (2003) are expected to have low direct impact on the development processes.

- **[C-8] Payment model**. The pricing models of Software-as-a-Service underlie the idea of variable usage based payments. Though different variations in the measurement of the usage exist, ranking from time-based measures like months of usage, to function-based consumption like service calls, all models have in common that there is no need for upfront investments of the customer in systems or licenses (Greschler et al. 2002). The process influence is expected to be rather low. A function-based measurement may require the software to be broken down into distinct units and therefore requires a certain modularity of the software with similar implications than discussed for characteristic C-6. In addition it would require a closer link between operations and billing functions.
- **[C-9] Ownership**. Turner et al. (2003) advocate the legal Software-as-a-Service characteristic of separating the possession and the ownership of the software. The consequences of this are however already included in the previous characteristics for example the limited customizing possibilities.

The analysis of the most cited characteristics of Software-as-a-Service solutions shows that some of them may challenge development processes and also offer new opportunities that may need certain process adjustment to be realized. [I-8] The service character of Software-as-a-Service solutions is a driver for continuous processes that do not end with the validation of the code and the shipment of the software to the customer. These processes also do not just include additional operations activities at the end of development as indicated in implication I-1, but constantly include the customer and his individual wishes as well as a permanent quality validation that delivers feedback about the customer's satisfaction, improvement ideas or the general execution quality back to the developers. The term 'development process' seems no longer appropriate since a close feedback cycles between operations and development functions are required.

Table 2 summarizes the found implications which form the basis for an evaluation of existing methodologies for software and especially service development. Characteristics with low expected process impact ([C-7] – [C-9]) were in the following not further examined.

Characteristics	Implications
[C-1]: Delivery: Process extension	[I-1]: Address execution-related activities
[C-2]: Delivery: Always current	[I-2]: Facilitate continuous development
[C-3]: Delivery: Central Infrastructure	[I-3]: Extend information sources [I-4]: Closer link between operation and development functions [I-5]: Restructure test and validation functions
[C-4]: Service character	[I-6]: Changed quality understanding [I-7]: Customer integration [I-8]: Continuous processes
[C-5]: Limited customizing	[I-7]: Facilitate customer specific requirements
[C-6]: Software: SOA	[I-9]: Handle / Enable modular software

Table 2: Summary of Implications

3 EVALUATION OF EXISTING METHODOLOGIES

For the evaluation of existing development methodologies against the derived implications, different approaches are collected from the software and service engineering literature and practice. These are then grouped by development artefact and process structure. Software development methodologies are divided into sequential, iterative and agile models. Service development methodologies are distinguished between sequential, parallel and continuous models. The group of combined software and service methodologies is not further subdivided. Table 3 summarizes the evaluation results with one established and well cited representative from each group.

As the summary shows, a direct application of current models in a Software-as-a-Service context is limited. Whereas the software development methodologies do not take service characteristics into account with the result of limitation in regard of deployment and customer interaction activities, the service methodology is offering only limited help in dealing with the complexity of software development. Software can be considered as part of the service potential dimension, which is included in nearly all service development methodologies, however the activities to obtain the potential are not further elaborated. Due to the generic character of the models and the vast variety of services, this is not surprising. The activities to provide the service infrastructure for a service offering of an airline can only hardly be compared with the once required for a translation agent. The models however try to be valid for both cases and thus are not able to provide a detailed activity breakdown. At this point a customization to the specific needs of the service under development is required.

A simple integration of current software development methodologies with service development methodologies with the aim to provide this kind of customization, e.g. the integration of the agile method extreme programming in the service methodology of Edvardsson et al. (1996), seems promising at first glance, a more detailed evaluation however reveals the danger of two distinct

requirements collection phases. This is caused by a very late start of potential securing activities in service methodologies which take place in or directly before the execution phase (Scheuing et al. 1989, Edvardsson et al. 1996). For Software-as-a-Service the service is however tightly coupled and entailed by software and its functionality. It is therefore necessary to link and embed the software requirements activities with the idea generating phase of services. The service potential dimension for Software-as-a-Service is represented by capacity and infrastructure providing activities. A delayed consideration of these may be possible, though there is a reasonable chance that infrastructure decisions affect and limit previously made decisions and should therefore also be taken into account in early requirements engineering phases.

The evaluated methodologies are very often focused or limited to development activities and therefore fail to satisfy implication I-1, the integration of execution related activities. The reasons for this trace back to the missing responsibilities of the vendors for the operation of the software. The look at customer-centric deployment methodologies that deal with the implementation at customer side could therefore be a promising approach to enhance the analysis.

On the development model level, the analysis reveals a few aspects of the existing models that are with good prospects to be of use in a Software-as-a-Service setting. Among these are the continuous development intention of agile methods, the quality assuring methods of Ramaswamy or the customer integration techniques of Edvardsson and Olsen.

Development model	Dev. artefact, process structure	[I-1]	[I-2]	[I-3]	[I-4]	[I-5]	[I-6]	[I-7]	[I-8]	[I-9]
V-Modell XT (Balzert 2008)	software, sequential	-	-	o	-	o	-	o	-	o
Rational Unified Process (Kruchten 1999)	software, iterative	+	+	o	o	o	-	o	o	+
Extreme Programming (Beck 2000)	software, agile	-	+	o	o	+	o	+	+	+
Scheuing und Johnson (Scheuing et al. 1989)	service, sequential	-	-	o	+	+	o	+	-	o
Edvardsson and Olsson (Edvardsson et al. 1996)	service, parallel	+	o	+	o	o	+	+	o	o
Ramaswamy (Ramaswamy 1994)	service, continuous	+	+	o	+	+	+	o	+	o
ServCASE (Meyer et al. 2008)	software & service	-	-	o	o	o	o	o	o	+

(+) model includes activities that address the implication;
(o) implication not specifically addressed in the model;
(-) realization of the implication not possible or excluded from the model

Table 3: Evaluation of Selected Development Models

Agile methods and their underlying mindset of short release cycles and emphasis on customer integration match with the derived implications of continuous development ([I-2]) and customer orientation ([I-7]). Problems may arise due to the error critical attribute of the multi-tenant architecture and the difficulties to scale the methods. Fry et al. (2007) however describe a successful application of agile methods at the Software-as-a-Service provider salesforce.com that was mainly driven by the wish for shorter cycle times.

In order to deal with the changed quality understanding ([I-6]), the model of Ramaswamy (1994) offers methods to link service ideas with customer value. During the development process, the methods in addition offer a toolset to connect these further with technical performance indicators of the software development or service level agreements (SLA). It therefore aims on the introduction of quality measurements that are operational and observable by the customer. Ramaswamy offers in addition one of the only models with a continuous process understanding that explicitly deals with activities of the service development as well as operations.

Edvardsson and Olsson (1996) state that beside an actual product dimension, other aspect like a marketing concept or the service potential have to be considered. They offer possibilities to integrate the customer in the whole process and regard it as important to include service potential securing activities early in the process.

4 CONCLUSION

The analysis of the potential effects of the Software-as-a-Service concept on the underlying development processes reveals a list of implications that in summary ask for a continuous process of development and operations actvities with a close linkage between the functions and high customer involvment. The collection of Software-as-a-Service characterisitcs that was extracted from definitions of Software-as-a-Service in current research papers, and that was used as the base for the analysis, summarizes the present Software-as-a-Service understanding and therefore creates a common ground for further discussions.

The service character that manifests itself by the importance of the external factors and the value creation through consumption, challanges the development-focused methodologies. The Software-as-a-Service marketing concepts furthermore pressure the release-based development and push for a continuous flow of enhancements. The centrality of the infrastructure within the control of the vendors also advances the information quality of development decissions by offering possibilities to evaluate user behavior within the system and analyse the quality of use of the offered solution. The Software-as-a-Service concept in addition offers seamless integrations of collaboration tools that can further intensify the interaction with the customer.

The current methodologies are only limitedly suitable to deal with the new challanges and opportunities. An extention by service specific requirements and operations activities are certainly neccesary. A simple integration of software and service methodologies or a connection of development and deployment models are however not recommended. The lower standarization options of services due to the involvement of the customer in the processes and the resulting customizing requisitions reveal a research demand for methods that enable standarized but flexible solutions. The collected implications demonstrate that a distinct treatment of software and service aspects are not appropriate in the Software-as-a-Service context anymore. Companies and researchers have to integrate the two domains into one joined methodology for Software-as-a-Service solutions.

In order to increase the validity of the results and enhance the research methodology, case studies in companies with software, service and Software-as-a-Service offering are planned to be conducted in the future. The further research objective is to form a comprehensive methodology for Software-as-a-Service projects, taking the derived implications and current practices in software companies into consideration and being able to offer recommended actions for practitioners. In addition, the scope will be broadened to include deployment models and therefore methodologies of customers, whose process responsibility has shifted to the vendors.

References

Aulbach, S., Grust, T., Jacobs, D., Kemper, A. and Rittinger, J. (2008). Multi-tenant databases for software as a service: schema-mapping techniques. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (Wang, J. Ed.), June 9 – 12, Vancouver, BC, Canada, ACM, 1195-1206.

Balzert, H. (2008). Lehrbuch der Softwaretechnik – Softwaremanagement. 2nd Edition. Spektrum, Heidelberg.

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J. and Slaughter, S. (2003). Is internet-speed software development different?. IEEE Software, 20 (6), 70-77.

Beck, K. (2000) Extreme programming explained: Embrace change, Addison-Wesley, Reading, MA, USA.

Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay, L. and Munro, M. (2000). Service-based software: The future for flexible software. In Proceedings of the Seventh Asia-Pacific Software Engineering Conference (Titsworth, F.M. Ed.) December 5 – 8, Singapore, 214-221.

Böttcher, M. and Meyer, K. (2004). IT-basierte Dienstleistungen. In Entwicklung IT-basierter Dienstleistungen in der Praxis (Fähnrich, K. and van Husen, C. Eds.). Frauenhofer IRB-Verlag, Stuttgart, 10-20.

Brehm, L., Heinzl, A. and Markus, L. (2001). Tailoring ERP systems: A spectrum of choises and their implications. In Proceedings of the 34. Hawaii International Conference of System Science (HICSS) (Nunamaker, J. and Sprague, R. Eds.), January 3 – 6, Maui, Hawaii.

Brereton, P., Budgen, D., Bennett, K., Munro, M., Layzell, P., Macaulay, L., Griffiths, D. and Stannett, C. (1999). The future of software. Communications of the ACM, 42 (12), 78-84.

Buxmann, P. and Hess, T. (2008). Software as a Service. Wirtschaftsinformatik, 50 (6), 500-503.

Choudhary, V. (2007). Comparison of software quality under perpetual licensing and software as a service. Journal of Management Information Systems, 24 (2), 141-165.

Cusumano, M.A. (2008). The changing software business: Moving from products to services. Computer, 41 (1), 20-27.

Darden, L. (1991). Theory change in science. Oxford University Press, New York.

Edvardsson, B. and Olsen, J. (1996). Key concepts for new service development. In The Service Industries Journal, 16 (2), 140-164.

Erl, T. (2008). SOA – Principles of Service Design. Pearson Education, Boston.

Espadas, J., Concha, D. And Molina, A. (2008). Application Development over Software-as-a-Service platforms. 2008 The Third International Conference on Software Engineering Advances (ICSEA) (Mannaert, H., Ohta, T., Dini, C. and Pellerin, R. Eds.), October 26 – 31, Sliema, Malta, 97-104.

Fry, C. and Greene, S. (2007). Large scake agile transformation in an on-demand world. In Proceedings of the AGILE 2007 (Eckstein, J., Maurer, F., Davies, R., Melnik, G. and Pollice, G. Eds.), August 13 – 17, Washington, DC, USA, 136-142.

Greschler, D. and Mangan, T. (2002). Networking lessons in delivering ‘software as a service‘ – part I. International Journal of Network Management, 12 (5), 317-321.

Heinrich, L., Heinzl, A. and Roithmayr, F. (2004). Wirtschaftsinformatik - Lexikon. 7th Edition. Oldenbourg, München.

Herder, E. (2007). An analysis of user behaviour on the web – Understanding the web and its users. VDM Verlag, Saarbrücken.

Kruchten, P. (1999). Der rational unified process. Addisson-Wesley, München.

Lassila, A. (2006). Taking a service-oriented perspective on software business: How to move from product business to online service business. In IADIS International Journal on WWW/Internet, 4 (1), 70-82.

Lindholm, K.R. (2007). The user experience of software-as-a-service applications. In 2007 Information & Service Design Symposium, March 2, Berkeley, USA.

Meyer, K. and van Hussen, C. (2008). Die ServCASE-Methode im Überblick. In Entwicklung IT-basierter Dienstleistungen (Fähnrich, K. and van Hussen, C. Eds.). Physica-Verlag, Heidelberg, 11-25.

Olsen, R. (2006). Transitioning to Software as a Service: Realigning software engineering practices with the new business model. In 2006 IEEE International Conference on Service Operations and Logistics, and Informatics, June 21 – 23, Shanghai, China, IEEE, 266-271.

Ramaswamy, R. (1994). Design and management of service processes: Keeping customers for Life. Addison-Wesley, Reading, MA, USA.

Sääksjärvi, M., Lassila, A. and Nordström, H. (2005). Evaluating the software as a service business model: From CPU time-sharing to online innovation sharing. In Proceedings of the IADIS International Conference e-Society 2005 (Isaisas, P., Kommers, P. and Mc-Pherson, M. Eds.), June 27 – 30, Qawra, Malta, 177-186.

Saeed, M. and Jaffar-Ur-Rehman, M. (2005). Enhancement of software engineering by shifting from software product to software service. In First International Conference on Information and Communication Technologies (ICICT 2005) (Khan, W. and Ahmed, F. Eds.), August 27 – 28, Karachi, Pakistan, 302-308.

Scheuing, E.E. and Johnson, E.M. (1989). A proposed model for new service development. In The Journal of Services Marketing, 3 (2), 25-34.
Sommerville, I. (2007). Software engineering. 8th Edition. Pearson Education, Harlow.
Sun, W., Zhang, K., Chen, S., Zhang, X. and Liang, H. (2007). Software as a Service: An integration perspective. In Service-Oriented Computing – ICSIC 2007 (Krämer, B., Lin, K. and Narasimhan, P. Eds.), September 17 – 20, Vienna, Austria, Springer, 558-569.
Susarla, A., Barua, A. and Whinston, A. (2009). A transaction cost perspective of the "software as a service" business model. Journal of Management Information Systems, 26 (2), 205-240.
Turner, M., Budgen, D. and Brereton, P. (2003). Turning software into a service. Computer, 36 (10), 38-44.
Xin, M. and Levina, N. (2008). Software-as-a-Service model: Elaborating client-side adoption factors. In Proceedings of the 29th International Conference on Information Systems (Boland, R., Limayem, M. and Pentland, B. Eds.), December 14 – 17, Paris, France.
Yadav, M.S. (2010). The decline of conceptual articles and implications for knowledge development. Journal of Marketing, 74 (1), 1-19.
Young, L. (2008). From Products to Services: Insight and experience form companies which have embraced the service economy. John Wiley & son, Chichester.
Zencke, P. and Eichin, R. (2008). SAP Business ByDesign – Die neue Mittelstandslösung der SAP. Wirtschaftsinformatik, 50 (1), 47-51.