Impact on Requirement Elicitation Process when Transforming Software from Product Model to a Service Model

Sameen Fatima, Amna Anwer[1†] and Adil Tareen[2††],
Sameenfatima35@gmail.com, amnanawer70@gmail.com , adiltareen123@gmail.com
University of Engineering & Technology, Lahore, Pakistan[1†], University of Management & Technology, Lahore Pakistan[2††]

Abstract
Influential trend that widely reflected the software engineering industry is service oriented architecture. Vendors are migrating towards cloud environment to benefit their organization. Companies usually offer products and services with a goal to solve problems at customer end. Because customers are more interested in solution of their problem rather than focusing on products or services. In software industry the approach in which customers' problems are solved by providing services is known as software as a service. However, software development life cycle encounters enormous changes when migrating software from product model to service model. Enough research has been done on the overall development process but a limited work has been done on the factors that influence requirements elicitation process. This paper focuses on those changes that influence requirement elicitation process and proposes a systematic methodology for transformation of software from product to service model in a successful manner. The paper then elaborates the benefits that inherently come along with elicitation process in cloud environment. The paper also describes the problems during transformation. The paper concludes that requirement engineering process turn out to be more profitable after transformation of traditional software from product to service model.

Keywords:
Software as a Service (SaaS), Requirement Engineering process, Requirement elicitation

1. Introduction

According to a study only 20% of IT corporations prefers software as service and consider this process very important. Whereas, majority of companies consider it to be average because of security, availability of system, rate of performance and integration with existing systems. A study shows that providing software as a service model leads to a profit margin of 45% per year. Software as a service (SaaS) is analysed as a software that is accessible over internet and is hosted by the software provider. Provider charges fee on regular basis for software use. Whereas, Software as a product (SaaP) solutions are those in which customer has to buy the license to use the solution. SaaP solutions are hosted by customer himself [2].

SaaS mainly focuses on distinguishing owner and customer of a software as separate entities. The tools, environment and other services that are provided by the provider are called platform as a service (Paas). The users of PaaS have no or very less control over the platform tools but they can fully control the deployed version of software applications. Another element of cloud computing is Infrastructure as a service (IaaS) which is responsible for network services, storage facilities as well as other hardware mechanisms. Whereas, the customer is able to install and execute random software that may even include operating systems as well.

In SaaS model user data and software are stored and hosted on a central repository. In this case user do not buy the product itself but use software, its services and infrastructure as a rental facility and pay according to use of services [5]. There are many ways in which SaaP and SaaS differs from each other. SaaS owns a database and middleware oriented architecture, due to which nonfunctional requirements of this process differ from the traditional software product [1]. When migrating software to SaaS model the vendor must focus on change in architecture, requirements and overall process to make sure that transformation is successful [3].

This paper gives an analysis of differences between SaaP model and SaaS model and provide a systematic approach for successful migration.Section 2 of this paper involves the related work done in this field. Section 3 offered a proposed methodology for a successful transformation which includes a generic and systematic approach. Section 4 includes the case study which was conducted to validate the proposed solution. Section 5 and 6 describes the limitation and future work of our research.

2. Literature Review

According to the study of Bennett et.al [6] SaaS mainly focuses on human managing relationship between provider and customer. According to his study service based soft wares are highly flexible, user interactive and personalized due to which requirement elicitation process changes. The future work of his study focuses on essential changes in

Manuscript received August 5, 2023
Manuscript revised August 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.8.23

software development life cycle while transforming software to service based model.

According to study of Olsen et.al [5] SaaS model is different from SaaP in terms of vendor customer relationships. SaaS provides a long term relationship between customer and vendor also his study point out that upgrades in software must be non-disruptive.

M.P.Papazoglou [6] conducted study on traditional software architecture and according to his study design process need alteration when software is transforming from product model to service model. He was one of the very first authors who analyzed the effects of SaaS model on engineering and business processes.

According to the group study of Balian and Kumar [2] the engineering process and quality models that are adopted for SaaS are not enough for software transformation from SaaP to SaaS. Their study concludes that nonfunctional requirements need to be considered more seriously while developing software from scratch or moving from SaaP to SaaS.

Recent study of Tariq et.al [1] compared the software development process for SaaP and SaaS models. His study describes the impact of nonfunctional requirements on the SaaS development model. His study than concludes that due to change in engineering process number of stakeholders increases and a checklist should be made to keep record of new stakeholders. For this purpose, Capability Maturity Model Integration (CMMI) has been referenced by him for making checklist for new stakeholder.

Sufficient research has been done on the SaaP model and transformation of Service based systems to cloud based system. Cloud based systems are those systems, services or resources that are given to user on demand via internet. However, very limited work has been done on requirement elicitation process when transforming existing software into a service based model.

This paper suggests a systematic methodology and a generic approach for successful transformation of software from product model to service model. Moreover, it also covers the necessary changes that need to be accommodated in requirement elicitation process during transformation.

3. Proposed Methodology

Generic Approach:

This approach mainly emphases on the software development process for both SaaP and SaaS model. It compares the engineering process for both models and extracts the differences among them.

Table 1: Differences in requirement engineering process of SaaP and SaaS model

Software as a Product (SaaP)	Software as a Service (SaaS)
Limited number of stakeholders	Large number of stakeholders
Customer involvement is very little.	Customer involvement is very high
Version based customer relationship	Customer relationship is long term
Special survey techniques are used for customer feedback.	Customer feedback can be achieved directly from usage monitoring and updated on basis.
An upgrade is needed for feature enhancement	Feature enhancement can be done on regular basis without any delay.
Specific downtime is required for system update	Integration of update is easy and there will be no downtime, the process is almost seamless.
Updates have strong impact on the system sometimes it needs to be retrained.	Updates are on continuous basis and they do not have high impact on the system.
Testing and user acceptance is difficult.	Tested in modules according to updates and user accepted is achieved through grouping.
Bug fixing is scheduled.	Immediately fix bugs.

According to the comparison SaaP model involves limited number of stakeholders but in SaaS model variety of stakeholders are involved. According to research study of kumar[4] these stakeholders could be graphic designers, security experts, integration managers, requirement analysts, customers and marketing managers etc.

Along with the expansion of stakeholders SaaS requirement engineering process also have long term customer relationship among the user and the provider that makes customer involvement more strong than SaaP. Moreover, a direct feedback is required from the customer that can be achieved by monitoring usage of the system and motivating the end users to provide genuine feedback using feedback forms by making them realize that their involvement matters a lot in development and improvement of the software. The integration of new features and bug fixing in SaaP require downtime and needs to be scheduled before implementation but in case of SaaS integration of updates is seamless and less disruptive also bug fixing is immediate as the system is updated and tested iteratively.

SaaP sometimes needs retraining when upgraded but in case of SaaS upgrades are more frequent and smaller as compared to SaaP due to which it requires less training at the customer side.

Furthermore, as SaaS is hosted on a central repository acceptance testing of new features is easy in comparison to SaaP model. A proportion of particular user group is selected and features are rolled out to them for feedback. In some cases different versions of new features are rolled out among the different user groups and then the solution which is highly approved by the end user is implemented. The difference in engineering process of SaaP and SaaS and the above defined characteristics leads to an iterative development process as the development and updates are continuous in service oriented process along with the frequent feedback from customers. A systematic solution has been proposed to handle changes in requirement engineering process.

- **Systematic Approach:**

The most important and time consuming phase of the software development life cycle is requirement elicitation and analysis [figure 1]. While transforming software from SaaP model to SaaS model there is a huge change in requirement engineering mechanism. To accommodate these changes in software engineering process we need to handle new requirements iteratively.

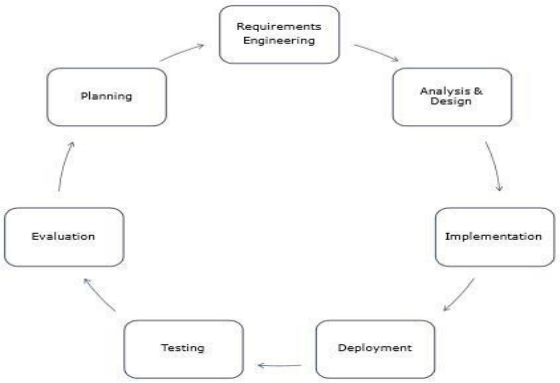

Figure 1: Iterative Software Development Mechanism

For a successful transformation of software from SaaP to SaaS the following steps of systematic approach needs to accommodate in requirement engineering mechanism.

1: As the requirement engineering process becomes iterative due to volatile requirements the development team needs to be trained for adaption of changes.

2: Requirement engineering process requires integration with iterative software development. Although iterative development process is not a distinctive property of SaaS model it also comes in SaaP model as well but the variable nature of requirements and frequent updates of system demands an iterative development model.

3: when shifting to SaaS model number of stakeholder increases. It is very important to identify right stakeholders during the requirement engineering mechanism. Identified stakeholders then needs to be prioritized according to a systematic approach.

4: As SaaS model demands strong customer involvement, so we can get customer attention by bug reports, feedbacks and sending themfeature requests invitations. End user should have a feeling that their interaction and feedback would be very effective for future updates and enhancements.

5: To get better understanding of requirements and to establish a better customer relationship feedback forms could be provided as well as usage of the software could be monitored to analyze that how system is behaving on customer end and which requirements need modification in engineering mechanism.

6: Updates integration should be seamless. One of the main benefits of service based software is that updates are so frequent and developers have full control over the deployment of the software. Due to frequent and small updates, downtime of the system turns out to be almost zero and database amendments are too small to affect the client side working. Furthermore, updates are installed on the server and customer has nothing to do with the updates unlike a traditional software product.

7: For acceptance testing of the software make multiple versions of the software update and roll them out among the various user groups. Feedback forms and usage monitoring techniques could be used for this purpose.

CASE STUDY

To validate our proposed methodology we have conducted a case study of EHR (Electronic Health Record) system of Ericsson Nikola Tesla (ENT) Corporation. This study is based on transformation of organization's centralized system to service based cloud software. Cloud software refers to the term cloud computing which in general is the distribution of hosted services over the internet [10].

Ericsson Healthcare Exchange (EHE) system was provided by ENT Corporation to the health care institutions and providers. EHR is the essential component of EHE system which stores all the patient, doctor, laboratory, medicine and research information of healthcare system. EHR is then responsible of sharing data among patients, health institutions, healthcare providers etc.

The existing version of EHR is not service oriented. Data is hosted on a central repository and accessible through direct database access.

However, due to some factors a service based cloud model is required for EHR.Those factors mainly are volatility and scalability in requirement engineering process as well as cost management. Furthermore, SaaS model is

required so that EHR is accessible for the institutions and patients who are living in remote areas.For the existing version of EHR it was very difficult to estimate the user defined requirements and suitable amount of resources. The tools and methods used for estimation were not cost effective especially in case of customized requirements by users. We have mapped our proposed methodology on the transformation of traditional EHR system to SaaS model and found the following results.

4. RESULTS

- **Stake Holder Identification:**

The transformation of EHR to SaaS model identified an increase in stakeholders. The following list of stakeholders has been found and identified during the requirement elicitation process.

1: System architect

2: System Builder

3: Service Provider

4: Service Consumer

5: Product Manager

6: System Engineer

7: Performance Analyst

1. Iterative Process:

The process was handled iteratively because of new services and customer specified requirements. And after every service deployment requirement were re checked to make sure that they are implemented properly.

2. Customer Relationship:

To make sure that customers are getting secure access of services continuously, services that were being used by remote areas customers were monitored. Feedback forms were created for users to get an idea of processing speeds and new service requirements.

3. Trained Staff:

As the process of requirement elicitation is iterative for that performance analyst, system engineer and service providers were hired for full time monitoring of the system.

4. Seamless Update:

In EHR system users didn't had to worry about the upgrades and updates. All the updates and upgrades were handled by the hosting server and were integrated to the system using application programming interface (API). It was performance analyst's job to make sure that enough resources are available to avoid downtime of the system.

5. Acceptance Testing:

As the system was developed at a national level the users of the system were known. Different user groups were made to gain acceptance of the new or modified feature. Emails were forwarded to the specified user groups depending upon the service to be developed.

Limitations :

This research work only focuses on requirement engineering mechanism of software development life cycle and do not cover all the phases due to which implementation of this methodology could be failed when considering all phases.

The other limitation of this research work is that it mainly emphasizes on the web based service model. For other SaaS models transformations, it might not be the very good approach.

5. Conclusion & Future Work:

During the process of transformation to SaaS model software development engineering process encounter changes radically which make requirement engineering process volatile unlike the process of traditional software.

Variation in process involves stakeholders, Strong customer involvement and their long term relationship management, more frequent upgrades and feature improvements and resource utilization for achieving service oriented goals.This paper has proposed a systematic solution for a successful transformation of a traditional product to service based model. This approach mainly focuses on necessary changes that need to adapt in requirement engineering mechanism during the migration.

The Future vision of this research is to study that how this new proposed requirement elicitation methodology can be integrated with agile development process. Which is already incremental and iterative in nature, and what will be the impact of this new method on the agile development engineering process.

6. References

[1] A. Tariq, S. A. Khan, and S. Iftikhar. Requirements engineering process for software-as-a-service (saas) cloud

environment. In Emerging Technologies (ICET), 2014 International Conference on, pages 13–18. IEEE, 2014.
[2] N. Baliyan and S. Kumar. Towards software engineering paradigm for software as a service. In Contemporary Computing (IC3), 2014 Seventh International Conference on, pages 329–333. IEEE, 2014.
[3] M. A. Chauhan and M. A. Babar. Migrating service-oriented system to cloud computing: An experience report. In Cloud Computing (CLOUD), 2011 IEEE International Conference on, pages 404–411. IEEE, 2011.
[4] S. Kumar and S. Sangwan. Adapting the software engineering process to web engineering process. International Journal of Computing and Business Research, 2(1), 2011.
[5] E. R. Olsen. Transitioning to software as a service: Realigning software engineering practices with the new business model. In Service Operations and Logistics, and Informatics, 2006. SOLI'06. IEEE International Conference on, pages 266–271. IEEE, 2006
[6] M. P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on, pages 3–12. IEEE, 2003.
[7] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, and M. Munro. Service-based software: The future for flexible software. In Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia- Pacific, pages 214–221. IEEE, 2000.
[8] "Cloud computing," in Wikipedia, Wikimedia Foundation, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Cloud_com puting. Accessed: Jan. 22, 2017.
[9] A. Gorokhova, "Software as a product vs. Software as a service," https://www.facebook.com/GetBynder,2016. [Online].Available:https://blog.bynder.com/en/knowledge/our-log/software-as-a-product-vs- software-as-a-service. Accessed: Jan. 22, 2017.
[10] Posted and M. Rouse, "What is cloud computing? - definition from WhatIs.com," SearchCloudComputing, 2012. [Online]. Available: http://searchcloudcomputing.techtarget.com/definition/cloud-computing. Accessed: Jan. 22, 2017.