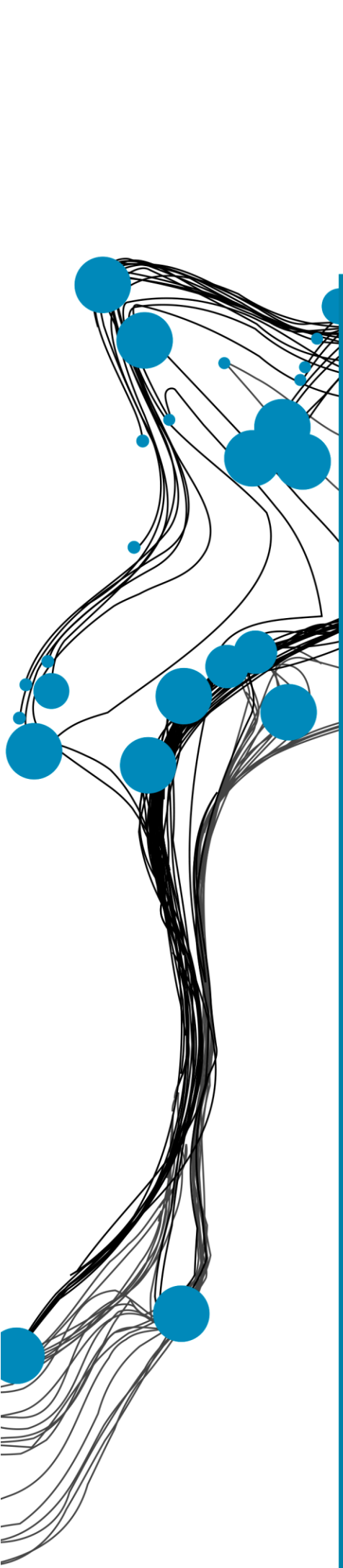

MASTER THESIS

DESIGN OF AN IMPLEMENTATION METHOD FOR CUSTOMIZABLE SAAS SOLUTIONS

L.G. Sterrenburg

APRIL 2019

COFANO

UNIVERSITY OF TWENTE.

MASTER THESIS

Design of an Implementation Method for Customizable SaaS Solutions

April 2019

Author

Name:	L.G. Sterrenburg (Thierry)
Programme:	MSc Business Information Technology
Institute:	University of Twente PO Box 217 7500 AE Enschede The Netherlands
Faculty:	Electrical Engineering, Mathematics and Computer Science (EEMCS)
E-mail:	l.g.sterrenburg@alumnus.utwente.nl

Graduation Committee

prof. dr. Maria-Eugenia Iacob

UNIVERSITY OF TWENTE.

Department:	Industrial Engineering and Business Information Systems
E-mail:	m.e.iacob@utwente.nl

dr. ir. Marten van Sinderen

UNIVERSITY OF TWENTE.

Department:	Computer Science
E-mail:	m.j.vansinderen@utwente.nl

Leon de Vries

COFANO

Company:	Cofano Software Solutions B.V.
E-mail:	leon.devries@cofano.nl

PREFACE

This research is my master thesis that completes my master study 'Business Information Technology' at the University of Twente. As master student I got in contact with diverse companies with interesting business cases. I appreciate all the experiences I have gained in the study.

This research is done in cooperation with Cofano Software Solutions. During the research I also worked on multiple implementation projects of customizable SaaS. Due to the projects, I gained more in depth experience in these projects, which helped me in this research. Our mutual aim was to define a standard and structured method for implementing customizable SaaS in the primary business processes. The designed method should also be applicable for implementation projects of Cofano's software.

First, I would like to thank my supervisors Maria Iacob and Marten van Sinderen for their valuable feedback and support. I appreciate the discussions that we had during the research. Your guidance sharpened this research.

Secondly, I would like to thank Leon de Vries for supervising me on behalf of Cofano Software Solutions. Thanks towards Marco Huijsman and Koos Leeuwenstein for providing a great work environment for performing this research. Also your interest in this research is much appreciated. Besides I would like to thank all my colleagues that have cooperated in the implementation projects and in this research for their effort and for sharing their experiences.

Finally, I would like to thank my family and especially my girlfriend Marlies for supporting me during my study and master thesis.

I hope that you will find in interesting and enjoyable to read this research.

EXECUTIVE SUMMARY

Software as a Service is a relative new business model for software to a customer. Often there is a periodic fee for using the software. Support and updates of the software are included in the subscription. For customizable SaaS, the customers play an active role in improving the software. A downside is that customers handle the SaaS solution as software on demand.

Implementing software in the primary businesses process of a business is critical project. The essential value of a service or product is gained in the primary business processes. It is of major importance that new software solutions support those primary business processes and that the software solution is adapted well by the business.

Currently there is no implementation method for implementating customizable SaaS in the primary business processes of enterprises. Hence, no suitable controls are available for these implementations projects. This has resulted in longer implementation projects than estimated. In the end this can lead to unsuccessful implementations, what can be prevented by having an implementation method.

This research has a research goal:

Develop and validate a method for implementation of SaaS solutions in the primary business processes.

In this research a SLR is performed for finding literature that connects with the research goal. After performing the SLR, it could be concluded that there is no implementation method for customizable SaaS solutions. Therefore a method for implementing customizable SaaS in the primary business processes has been developed and validated in this research.

The most important results of this research are:

- In the literature no implementation method for customizable SaaS in the primary business process has been found.
- A method for implementing customizable SaaS in the primary business processes has been developed in this research.
- The steps in the implementation method are based on concepts found in literature and field experiences.
- The developed implementation method has been validated by experts and their opnion towards the implementation method is very positive.
- The designed implementation method will be used in practice for Cofano's implementation projects.

Table of Contents

Acronyms

ASP – Application Service Provider
BPMN – Business Process Model and Notation
BPR – Business Process Reengineering
CIB – Cloud Interoperability Broker
EROP - Events, Rights, Obligations and Prohibitions
IaaS – Infrastructure as a Service
LP – Local Business Policy
MVP – Minimum Viable Product
PaaS – Platform as a Service
ROI – Return on Investment
PM – Policy Manager
QoS – Quality of Service
SaaS – Software as a Service
SA – Service Agreement
SLA – Service Level Agreement
SLR – Structured Literature Review
SPL – Software Product Line

1 Introduction

Software as a Service (SaaS) is a relative new way of providing software to a customer. The business model behind SaaS solutions is that the customer does not have to buy the software, but instead pays a periodic fee. The SaaS provider is responsible for the deployment and maintenance of the software in the cloud. Therefore, customers do not have invest in IT-hardware for installing and running on premise applications. SaaS applications have a low total cost of ownership in comparison with on premise-installed applications. Where on premise-installed applications are often configured once, SaaS applications have periodic updates, which can be optionally deployed. This strategy ensures that the software stays up-to date and consequently does not age (fast). The down-side of these updates is that the customer sees the software as an on demand service. Customers don't have to make an internal request for change, because no extra costs will be charged. They just send a request to the ASP (Application Service Provider) and expect it to be implemented within a short timeframe. As a result a growing number of customers results in a growing number of implementation requests.

1.1 Business Models for Software

The business model for customizable SaaS applications differs from the traditional model for on premise installed applications. Due to the difference in business model, the development of the software and deployment of the software are organized in another way. This has also an impact on the approach for implementing the software at customers.

On premise software is installed on the location or in a cloud environment. When the software is installed on the customer location, the hardware is managed by the customer. The initial costs for on premise software are high compared with the initial costs for SaaS. The software is often a basic off the shell package whereon additional modules can be installed. Besides it is possible to realize all sorts of customizations. The reason for the high initial costs are that customizations are realized and used for one customer only. Reuse of code is not applied much since those customizations are very specific requests and built for a particular version of the standard package. Updates of on premise software are only applied when necessary, because these updates come with additional upgrade costs and also result in a period of down-time.

The business model for SaaS applications is different. The initial costs are lower than for on premise software. Usually a starting fee has to be paid and thereafter there will be a monthly fee. This periodic fee includes the hosting of the software accompanying with periodic updates. All customers will use an application with the same code base. Therefore features have to be built once and can be provided to all customers. Customers are able to make requests for features, which are reviewed by the ASP and may end up on the roadmap. Before accepting and implementing a feature it is important to examine the impact on the application. The implementation or adaption of a feature will have an impact for all customers, since they share the same code base. In order to accept and support different views on the business operations, configuration options and settings can be used to make the application fit for different customers.

1.2 Barriers for SaaS Solutions

Where on premise installed software is completely in control of the customer, SaaS applications are managed by the ASP. Besides the advantages that the customer does not have to manage and maintain the IT infrastructure also raises barriers. Weaknesses of cloud solutions are reliability, limited customization, limited customizability and no dedicated personnel. (Bibi, Katsaros, & Bozanis, 2012).

The application is maintained and managed by the ASP on a remote location. This might raise reliability issues and could cause data loss. In case of application restarts or network interruptions the application is offline for a certain time. On premise software runs locally and will only be offline when the local network or server does are not functioning correctly. Due to the business model of SaaS applications there is limited customizability and limited configurability, because all customers have software with the same codebase. However, when needed the ASP can implement configuration functions in order to meet all customizations and configuration requirements. Other mentioned downsides are:

- Data confidentially, integrity and availability
- Legal problems from cross-country distribution
- No clear downtime agreements or reimbursement policies.

1.3 Primary Business Processes

Business processes are activities that add value to a business product. According to Porters' value chain there are primary activities and support activities. The idea is that a product gains value with each activity. The sum of all gains from the activities is the margin that is gained at the end of the value chain. The primary activities contain the core business, where the actual product or service is delivered. The primary activities are explained below.

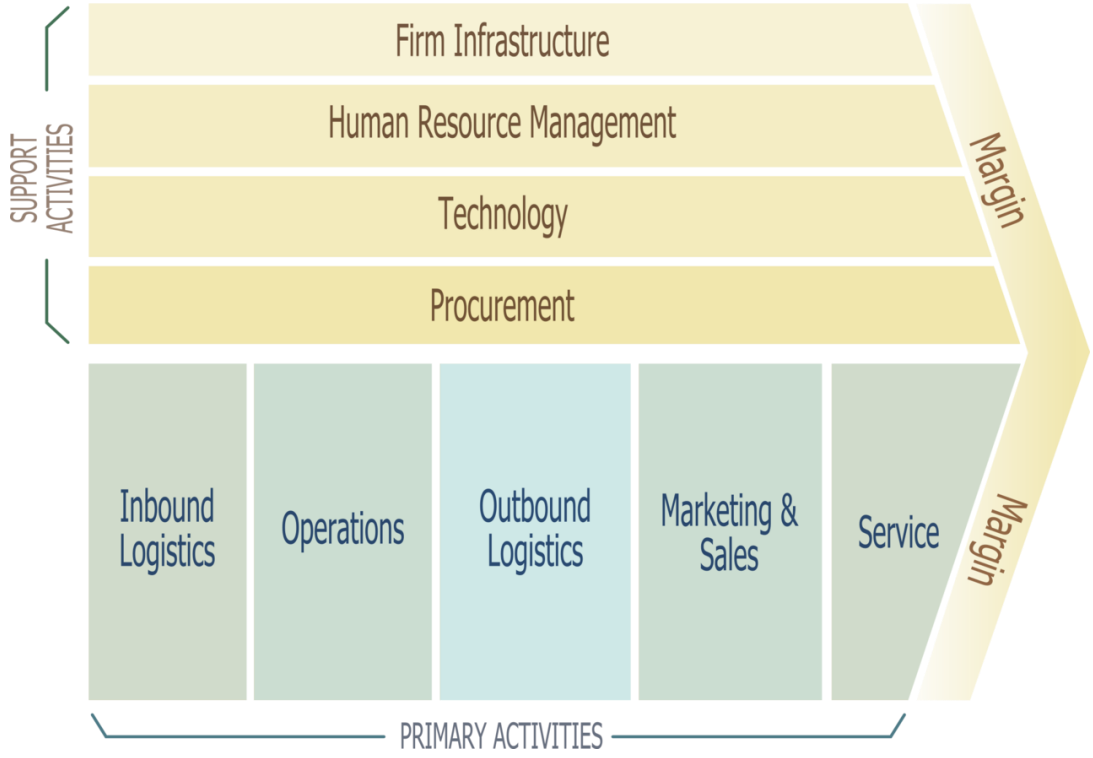

Figure 1: *Porter's Value Chain*

Primary Activities

- Inbound Logistics: All incoming movements of materials from suppliers
- Operations: Activities needed to transform the input to an output.
- Outbound logistics: All outgoing movements of the final product to customers or retailers.
- Marketing & Sales: Activities needed for promoting and selling the service or product.
- Service: This activities concerns the service and aftersales.

Businesses that are located in the tertiary sector only offer one or multiple services to their customers. For these businesses, there are no inbound or outbound logistics, because they do not manufacture products. Implementation of software in the primary activities of a business requires a different approach than the implementation for supporting activities. The primary activities ensure that a product or service is delivered in the end. When software for controlling these activities is not implemented correctly, less or no value is added in the end.

1.4 Use Case Context

Cofano Software Solutions B.V. is an ASP of diverse products for managing business processes and logistics. All Cofano's services are provided as Software as a Service. One of their logistic applications is STACK. It is an application for inland terminals, forwarders and chain directors for managing container transport. The application supports in manual and automatic order entry for container transport. It contains modules for truck planning, barge planning and train planning to provide full management for all usable modalities in the container transport. Also apps are built to enable hands on registration of actions. An example of such an app, is the terminal app for reachstackers, which is used for registration of actions on the terminal. The finance module ensures that bookings can be invoiced. STACK supports in the whole flow from order to cash. The implementation strategy for STACK is used as object of study in this research. The implementation strategy entails the complete process starting presenting the application for the first time to an interested party. It also covers the fit gap analysis, development of features for realizing the minimum viable product and migration to the SaaS application. Currently there is no implementation method or strategy used. Implementation is done using common sense and by making mutual agreements between the ASP and the customer. This has resulted in longer implementation processes than estimated and pre-arranged. Due to the maturing of the application, the number of interested parties rises. Along with the raising interest, there is also a rise in customers using the product and businesses where the implementation proces has started.

1.5 Research Goal

This research investigates the implementation of a SaaS application in the primary business processes of enterprises. Several enterprises are currently using STACK. The implementation and migration phase for these enterprises have been diverse. Besides there was also an implementation project that has been stopped after exceeding the estimated implementation time. In order to mitigate the risks of implementing a SaaS application, an implementation method that allows control over the implementation process has to be developed. We intend that the designed method can also be used for the implementation of other SaaS applications. The following research goal is defined:

Develop and validate a method for implementation of SaaS solutions in the primary business processes.

1.6 Research Questions

The following research questions must be answered in order to achieve the research goal:

1. How are SaaS solutions implemented in the primary processes of enterprises?
2. Which steps can be identified in the process of migration to a SaaS solution?
3. How can the designed implementation method be validated?

1.7 Research Methodology

The research methodology used in this report is Design Science Methodology (DSM) developed by Wieringa (Wieringa, 2014). The methodology is used as guide for designing an implementation method for customizable SaaS applications.

1.8 Structure Report

The report is structured as follows: Chapter 2 outlines the problem statement and describes the literature selection process. Chapter 3 contains an analysis of the selected literature followed by an overview of the analysis in chapter 4 . Chapter 5 shows the design of an implementation method that can be used for implementing a SaaS application in the primary business processes. The implementation method is validated in chapter 6. Chapter 7 concludes the findings in this report, including recommendations for future research and for the use of the designed method.

2 Research Design

In this research a systematic literature review (SLR) will be used. This SLR give insight on available existing literature and knowledge gaps. The method for performing the SLR is described in section 2.1. Section 2.2 presents the problem statement for the SLR, followed by the research questions in section 2.4. Based on the research questions a search process is established in section 2.5. For the right data selection, exlusion criteria are defined, which can be found in section 2.6. The process for data collection is shown in section section 2.7. An overview of the results in shown in section 2.8.

2.1 Method

In order to perform a SLR a defined method is needed. Brereton and Budgen (Budgen & Brereton, 2006) defined characteristics for a systematic review. These characteristics are used for performing a SLR in this research. The following characteristics are defined by Brereton and Budgen (Budgen & Brereton, 2006):

- review protocol
- defined search strategy
- documented search strategy
- inclusion and exclusion specification
- evaluation of obtained information

Besides the characteristics presented above, Brereton and Budgen (Budgen & Brereton, 2006) propose a process with three phases for reviews in Software Engineering:

- Planning
- Conducting
- Reporting

The proposed process is used in this research. The planning phase is performed in the search process, where literature is searched that is connected with the topic. After defining the right search queries, the set of results will be reviewed. The review of the results can be mapped on the conducting phase. Reporting will is done in the section 'Discussion' where the relevancy and fit and gaps of the literature will is discussed.

In "*Systematic literature reviews in software engineering - A systematic literature review*" (Kitchenham et al., 2009), Kitchenham et al. perform a systematic literature review. The SLR by Kitchemham et al. is an example for performing a SLR. The planning phase is represented by the research questions, search process, exclusion criteria and quality assessment. Conducting the SLR is done in the subsection 'Data Collection'. The last phase reporting is represented by the section results, which is followed by a detailed analysis.

2.2 Problem Statement

The business model of SaaS solutions is that there is main code base that can be sold as a service to customers for a periodic fee. The impact for the use of the software depends on the complexity of the software and the effected business processes. For there is one code base, all (potential) s have to find a fit for their processes in the software or the software should be extended with new functions. Besides processes can be made more efficient with new software, but in order to achieve this, processes may have to change in order to become more efficient.

Problem Statement: How can SaaS-based enterprise solutions implemented successful in the primary business processes?

2.3 Literature Research Questions

The literature research questions should help in finding existing literature about how implementation of customizable SaaS is done. The keyword ‘enterprise’ and ‘business’ are exchangeable. However, using ‘enterprise’ as keyword result in more literature.

- RQ1 How are SaaS solutions implemented in enterprises?
- RQ2 How do client and provider cooperate in building customizable SaaS?

2.4 Search Process

The search process shows how the SLR is performed. The found literature should give an insight in what literature exist regarding the implementation of customizable SaaS in the primary business processes. In order to know how software and in particular a SaaS solution is implemented, the starting point of the search process was a broad scope, which was narrowed down to keep only the most relevant literature results. The main topic are split into four sections:

- implementation
- customizable
- SaaS
- primary business processes

For the part 'customizable' there are similar words, f.e. 'tailoring' or 'tailor-made' that are used in the branch of software engineering. SaaS is an abbreviation for Software as a Service, which may be used in literature instead of the abbreviation. The part of 'primary business processes' points where change of system will have an impact in the enterprise. However, these keywords might be to specific. Therefore enterprise software or enterprise applications might include relevant results, for enterprises is the target group for implementation. The keywords 'primary business processes' might be to specific for finding literature. Using 'enterprises' will be used as replacement for 'primary business process' in order to search with a broader scope.

Scopus and Google Scholar are used in the search process. Scopus is used as main search engine and the search queries are adjusted and improved for this search engine. Google Scholar is used as search engine to find interesting literature and explore usable key words. The key words found with Google Scholar are used for designing a search query in Scopus. The total number of results found in Google Scholar was to high even after using inclusion and exclusions for use in this SLR.

2.5 Exclusion Criteria

In order to get a good overview of the available literature the following criteria for exclusion are used:

- Duplicate articles, because Scopus can deliver one article multiple times as result.
- Books and book chapters, for they are not always available and the overall coverage of the books is not relevant in this SLR
- Unavailable articles, for Scopus also shows articles which are not accessable.
- Paid articles, for Scopus also shows results that are not freely accessable for the University of Twente.

2.6 Quality Assessment

The papers that are returned as result by Scopus are reviewed on the title and abstract. When an article is graded as irrelevant for this SLR, attention is paid to the key words. When a key word is considered irrelevant for this SLR, it will be added as an exclusion for the search query. In this way

effective filters are added to the search query. No further requirements are determined for grading an article on quality in order to keep the scope broad and to not miss any possible relevant information. In the discussion the quality of articles or statements in articles will examined on quality and validity.

2.7 Data Collection

For collecting the data three search queries are defined that cover the relevant data in this SLR:

- SaaS enterprise implementation
- customizable SaaS
- agile project management SaaS

The data collection for all three search queries are described below.

2.7.1 SaaS Enterprise Implementation

The first step in the searching process is designing a search query with all relevant key words. Since SaaS is the abbreviation for Software as a Service, this should be included in the search query. The following search query covers all results connected with the implementation of SaaS solutions in enterprises:

TITLE-ABS-KEY (("SaaS" OR "Software as a Service") AND ("enterprise software" OR "enterprise application") AND implementation) *{21 results}*

A total of 21 results were returned.

The search query is relative broad scoped, which means that all kinds of SaaS solution implementation for enterprises are returned. The results cover subjects such as "ERP as SaaS" and "cloud-computing". These subjects are not the core goal of the research, concluding that the search query should more focused on SaaS solutions as enterprise application or enterprise software.

Exclusion

All results with an undefined author were excluded from the search results. This brought the number of results to 17. In the results there appeared one book chapter, which was unavailable. Therefore book chapters where excluded as source type as well. Furthermore results were excluded due to irrelevancy. Results with the keywords: 'costs' and 'metadata' were excluded. The results from: 'Confenis 2013 7th International Conference On Research And Practical Issues Of Enterprise Information Systems' are held by another library and is for that reason excluded. After all exclusions 9 results were left over in total.

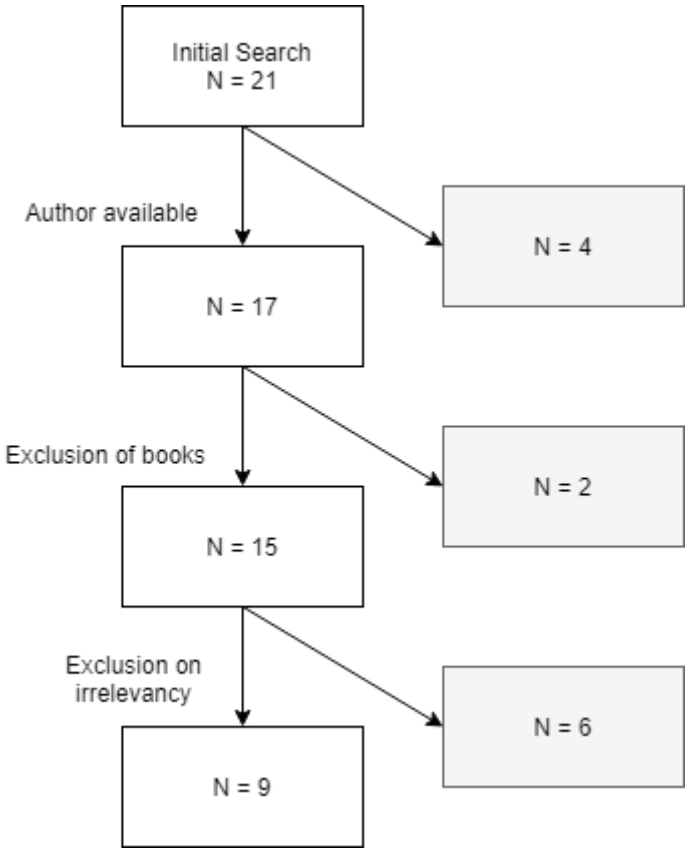

Figure 2: *Literature Selection Process for "SaaS Enterprise Implementation"*

Figure 2

2.7.1 Customizable SaaS

Besides the implementation of SaaS solution, the customisability of the solution is an important factor. Therefore a query is designed to retrieve results regarding the customisability of SaaS solutions.

TITLE-ABS-KEY (("customizable SaaS" OR "customisable SaaS" OR "customizable Software as a Service" OR "customizable Software as a Service")) *{9 results}*

Exclusion

With exclusion of the keyword 'XML' the irrelevant papers were filtered out. A total of 7 papers were selected as result for this search query.

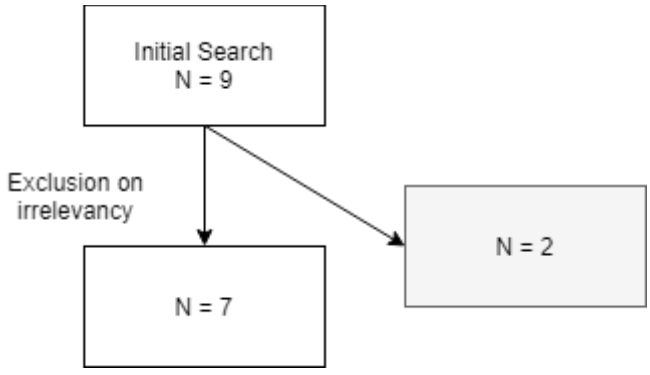

Figure 3: *Literature Selection Process for "Customizable SaaS"*

2.7.2 Agile Project Management

Having the implementation and customization of SaaS covered. Other results about the implementation of SaaS are found in the area of project managment. Therefore the following search query is used.

TITLE-ABS-KEY (("agile project management") AND (SaaS OR "Software as a Service")) *{14 results}*

Exclusion

After excluding irrelevant articles based on title and abstract 4 results were left.

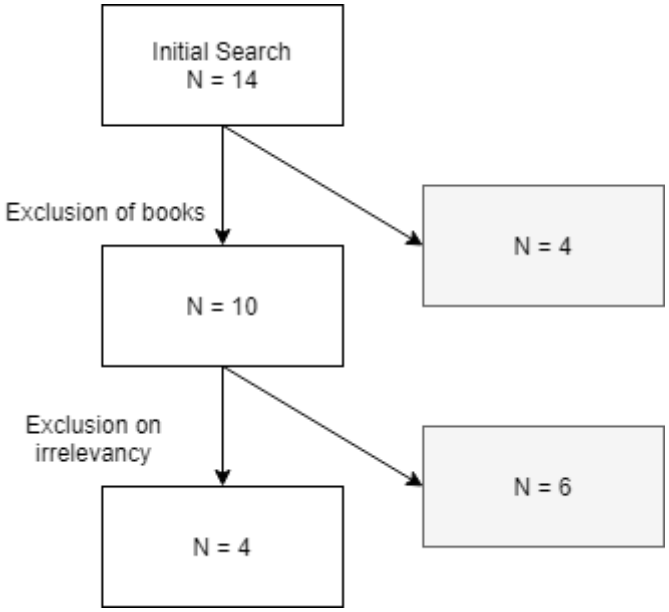

Figure 4: *Literature Selection Process for "Agile Proejct Management"*

2.8 Results

The found literature in the section 'Data Collection' is shown in table 1. The results are categorized on search query.

Search Query	Results	Total
SaaS enterprise implementation	(Ali et al., 2016), (Aulbach et al., 2008), (Buford et al., 2015), (Grati et al., 2015), (Koski et al., 2016), (Molina-Jimenez et al., 2011), (Moore & Mahmoud, 2009), (Singhto & Phakdee, 2017), (Seethamraju, 2015)	9
customizable SaaS	(Liu et al., 2010), (Moens et al., 2016), (Moens et al., 2012), (Moens & De Turck, 2014), (Mohamed et al., 2015), (Ruehl & Andelfinger, 2011), (Truyen et al., 2012)	7
agile project management SaaS	(Agarwal, 2011), (Bajighar & Shahzad, 2017), (Benefield, 2009), (Femmer et al., 2014)	4

Table 1: *Search Results*

3 Analysis

The retrieved results from the search queries in Scopus are analysed in this section. The found papers are categorized on subject. For each subject there is a separate section in this chapter. The analysis provides a global overview of the found literature in order gain insight in the available knowledge. The relevance and usability of the literature is examined in chapter 4.

Section	Subject	# Total	Papers
3.1	Change Management	1	(Seethamraju, 2015)
3.2	Software Development Methodology	1	(Singhto & Phakdee, 2017)
3.3	Data Migration	1	(Ali et al., 2016)
3.4	Multi-tenant Software	3	(Aulbach et al., 2008), (Liu et al., 2010), (Ruehl & Andelfinger, 2011)
3.5	Service Level Agreements	2	(Grati et al., 2015), (Koski et al., 2016)
3.6	Service Agreements	1	(Molina-Jimenez et al., 2011)
3.7	Business Rules	2	(Moore & Mahmoud, 2009), (Truyen et al., 2012)
3.8	Agile Project Management for SaaS	5	(Koski et al., 2016), (Benefield, 2009), (Bajighar & Shahzad, 2017), (Agarwal, 2011), (Femmer et al., 2014)
3.9	Customizable SaaS	4	(Mohamed et al., 2015), (Moens et al., 2012), (Moens & De Turck, 2014), (Moens & De Turck, 2016)

Table 2: *Categorization of the Papers per Subject*

3.1 Change Management

Change Management is an important part of the implementation of a SaaS solution. The software is hosted by the software vendor and the client can often login via a web browser or app. Seethamraju (Seethamraju, 2015) state that small and medium sized enterprises do not get a sense of ownership. Users would possibly not accept or use a new feature. The identified challenges for the adoption of a new SaaS solution are:

- Attidude towards the proposed change of the system
- Lack of process understanding
- Change in process steps
- New activities to be performed
- Replacement of resources
- Lose of control, access by updating the roles and permissions of people

In order to have a successful implementation the above mentioned challenges must be dealt with.

3.2 Software Development Methodology

For software development diverse methodologies can be chosen, from waterfall to extreme programming. Besides it is possible to use a hybrid development methodology, where only parts or a part from one or more methodologies are used for the development of the software. The characteristics of the (hybrid) methodology also determine the involvement of the product owner in the development and the software releases in the process. Singtho and Phakdee (Singhto & Phakdee, 2017) performed a case study concerning the development of tailor-made SaaS products for small and medium enterprises in Thai service and manufacturing. A hybrid solution is used as approach in this case study.

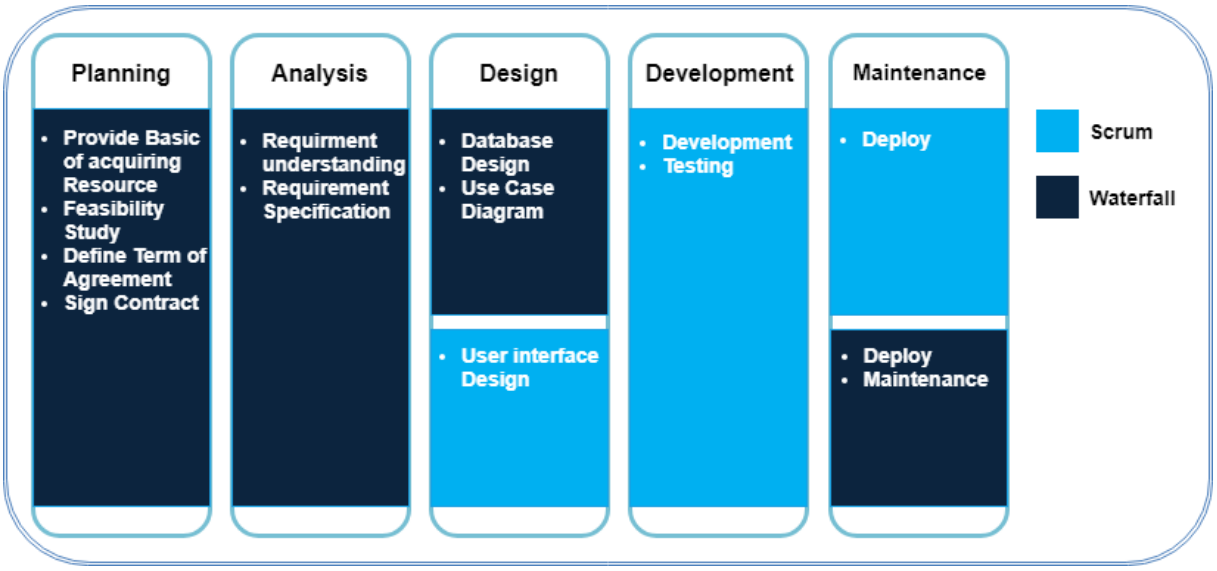

Figure 5: *Blending Scrum and Waterfall Technique (Singhto & Phakdee, 2017)*

3.3 Data Migration

When moving from one application to another data migration is often an important part of the transition. Data migration can be applied when the new application is similar of has partly similar features as the current used application. Ali et al. (Ali et al., 2016) explored the areas cloud-computing that would benefit interoperability by standardization. The focus in that research lies on data migration from SaaS to SaaS applications. A cloud interoperability broker (CIB) is proposed as a solution for data migration. The CIB should enable SaaS clients to move from one SaaS application to another. A methodology (Ali et al., 2016) is designed to for migrating data covering the following steps:

- Collection and Analysis of Metadata
- Develop the Mapping Model
- Solution Design
- Implementation
- Test the Solution

The collection of the metadata from both applications enables the CIB to develop a mapping model. The mapping mapping model ensures the right location for data in both applications is allocated. For each combination of applications a new mapping model has to be designed. The next step designing a solution for migrating the data with the help of the mapping model. Once the design is finished, the solution can be implemented and tested. If the tests failed, the mapping model should reviewed and the process should be resumed from the development of the mapping model. This process is repeated until the tests are successful.

3.4 Multi-Tenant Software

The advantage of SaaS is that there is a lower cost of ownership for the service client. Besides the service provider can offer the service on a shared system. An example of a shared system is a multi-tenant database (Aulbach et al., 2008). The suitability for using a multi-tenant database depends on the offered application as service. A simple e-mail application is able to offer a service to more tenants than an ERP application. Besides there are diverse ways to setup a multi-tenant database (Aulbach et al., 2008). It is possible to use private tables for each tenant, but tables can also be shared by using meta-data columns. The meta-data columns used are in minimal way a tenant and

row column. In this way tables for multiple tenants can be merged into one table. The downside is that there is an overhead of data, caused by the meta-data columns.

In multi-tenant systems enterprises are often the tenant and each tenant has multiple users that access the same data. Each tenant has their own processes and each process and be split up in services. In order to optimize the performance load balancing can be applied. Messages should be send to different priority queues (Liu et al., 2010).

Ruehl and Andelfinger (Ruehl & Andelfinger, 2011) designed an architectural model for customization of SaaS applications on the tenant level. Because the processes for tenants may differ, the application should be adapted for each tenant's process. With the use of software product lines, the application can be tailored for all tenants.

3.5 Service Level Agreements

The contracts of a SaaS solution often comes together with a service level agreement (SLA). Since the SaaS provider often use a PaaS or IaaS solution there is service build on cloud layers offered to the end user. Grati et al. (Grati et al., 2015) designed a model to manage entities in a layered cloud construction for SLAs. In the model a service can be composed by other services with underlying SLAs. These SLAs then have a direct effect on the SLA of the composed service, because of the dependency. The management of these entities is important of SLA violation and possible penalties. SLAs are always established between two parties. For SaaS providers there are two SLAs, one between the SaaS provider and and the other SLA is established between the IaaS provider and SaaS provider. Service Level Monitoring can be done to prove that the agreements in the SLA are met. A service can have a certain minimum availability (for example 99.9%) excluding planned maintenance. (Koski et al., 2016) The following calculation can be used for monitoring the availability:

*uptime = 100 * (time-downtime)/time* (Schwaber & Beedle, 2001)

3.6 Service Agreements

Where the SLAs describe the agreed quality of service (QoS) between the client and provider, the service agreement describes what services are provided to the client. The rules or limits for using a service can differ per client, which appoints to a customisability for the SA. An example for implementing a customizable Service Agreement (SA) is presented by Molina et al. (Molina-Jimenez et al., 2011) In their proof of concept a policy manager (PM) is implemented between the gateway and the service interface. The policy manager, which checks the input given by the client on compliance with the SA. The result of this check on compliance can be either an acceptance or a refusal. Customisability of the SA can be done with local (private) business policies (LP). In the proof of concept both SA and LP are defined in the EROP (Events, Rights, Obligations and Prohibitions), which is a rule based contract specification language. This enables service providers to customize the SA for (a class of) clients.

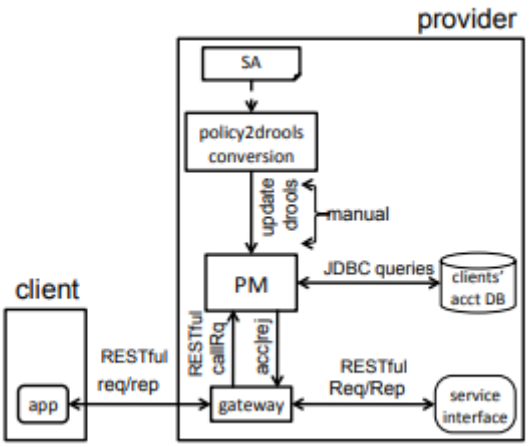

Figure 6: *Proof of Concept Implementation* (Molina-Jimenez et al., 2011)

3.7 Business Rules

Business rules define which information is needed and which actions are triggered by the given information. The process of defining business rules is needed to transform (sub)process into a service component. The designed service should be able to automate (a part of) the process. Moore et al. (Moore & Mahmoud, 2009) used a case study to show how business rules can be stated in words and transformed to web service definitions in WSDL.

Business rules also apply in context-oriented programming. In the same branch enterprises may have different business rules or processes, but are performing and delivering the same service. According to Truyen et al. (Truyen et al., 2012) customizability can be achieved be designing multiple objects, which can be used by tenants by dependency injection. As use case a system for online hotel bookings is used. Where hotels can configure their own tariffs, for low season, high season and VIP guests (Truyen et al., 2012). All tenants make use of the base layer can are able to add layers for customization of the software.

3.8 Agile Project Management for SaaS

One of the most important things for SaaS products is care. Since the application is not installed on-premises, but is just a service there is low total cost of ownership. When customers are unsatisfied with the service it is, economically seen, easily to switch between similar SaaS applications. Besides users are intended to complain more easily about the user-friendliness of the application. (Koski et al., 2016) The application might be used by several types of users. Some users will be professionals in the application, others might use only a minimum set of features. The application should be have an intuitive interface in order to gain a higher user satisfaction (Koski et al., 2016).

"Being wrong about what customers want can mean losing a when another service provider, with possibly an inferior solution by many aspects,hits the expectation target exactly." (Koski et al., 2016)

Customers see SaaS as an on demand service. Not only for the up-time of the service, but also for functionality development. Applying lean management on the deployment of the services might help in meeting the demand. Benefield (Benefield, 2009) describes four lean management techniques that can be applied on deployment strategies for SaaS:

- Poka Yoke
- Jidoka
- Kaizen

- Just in Time

Poka Yoke aims at mistake proofing. The release of new components for the software, and in special for long release cycles, may cause bugs. Variances of software environments should be kept to a minimum in order to be able to reproduce environments. Furhtermore mistakes and errors can be prevented by automated deployment and management tools. These tools can help in version control and connecting the right packages with a certain environment. When also test automation is realised, changes in the software can be tested, tracked and managed well. Jidoka follows the principle of automatically building artifacts that meausre and reports errors. This method helps in builiding more stable software, testing and troubleshooting. The Kaizen method can be used for continuous improvement. Not alone for solving errors in the system, but also for understanding the workflow of the customers. When tracking the activity of users it is possible to figure out which functions are popular and which not. Eventual the software can be improved in such way that the workflow of the can be improved by the software. The just in time method aims at the reduce of waste. If only functionalities are build that are really required, the waste is left out. Software architecture may add waste to the project, for a part of the flexibility is handed in and redesign may be necessary in the future.

Autonomous software deployment should help software providers in achieving continuous development of their software. According to Bajighar and Shahzad (Bajighar & Shahzad, 2017) this can be achieved by by the adoption of some fundamental principles for software development management. Since cycles for software releases becomes shorter it is important that developers are perpetual in development mode, because new features are build continuously. Post-agility is mentioned as possible next step in software development. Where post-agile is a combination of Waterfall or planned development and agile techniques. The following princples are described for self-driving software development:

- Every requirement is assigned a valuation
- Every requirement is assigned to a task-list
- Every task is assigned to a workflow
- Every team member is assigned to at least one role
- Every team member has a work queue
- Every requirements delivery status is available on-demand

The valuation of a requirement is done based on four factors, namely:work size, size of function, effort and return on investment (ROI). All task for delivering the requirement are stated in a task-list. For each task on the task-list there are workflows. All team members have at least one role assigned, and more roles are possible. The task-list can be split up assigned a task with a certain workflow a queue of a team member. In this way it is possible to monitor what the status of a requirement is. For the application of the principles of self-driving software application Bajighar and Shahzad (Bajighar & Shahzad, 2017) propose the following six gears:

- Rapid continuous design & requirements
- Rapid continuous development
- Rapid continuous feature assembly
- Rapid continuous testing & acceptance
- Rapid continuous marketing & training
- Rapid continuous product evolution

These gears ensure that every team member is continuously working on the development and evolution of the software.

Scrum as agile method can be used for short release cycles in software development. Especially Scrum type C allows to develop and release user stories fast (Agarwal, 2011).The concept of Scrum C, also known as continuous Scrum, is that each sprint has a release cycle of three weeks and each week a new sprints starts. This means the there are three sprints active simultaneously. Because there is every week a new release cycle and a new start of a sprint. User requests are picked up faster than in other types of Scrum. An use case at InstantApps has shown that with the use of Scrum C more work-items were finished in four release, than possible with other types of Scrum (Agarwal, 2011).

Due to the good connectivity and online collaboration tools, it becomes easier to set up distributed teams. Besides s that co-create the software can be geographically widespread, which can be a barrier for good transfer of requirement information. Femmer et al. (Femmer et al., 2014) designed an refined artifact model, based on an original model from previous research for organizing and managing software projects. The original model is used in a software project and during the project refined. The main artifacts of the refined model were: Planning, Requirement & Specification, Change Management and Testing. The original model was set up aiming at an agile way development. After refining the model it could be concluded that more traditional project management artifacts were added to the refined model. The artifacts from traditional project management were added, because more and more business context is needed when working with distributed teams.

3.9 Customizable SaaS

Customizable SaaS allows the software provider to serve multiple clients with diverse Software Product Lines (SPLs) within the same software package. A feature model can be designed for the application, where there can be mandatory, optional and alternative features. (Mohamed et al., 2015) The mandatory features belong to the core of the application and are required to let the software function. The optional features are additions to the core features to support or automate a part of the business process. Alternative features contain functionality for similar processes, but have a slight other type of content. For example products can be bought in a webshop by credit card or other online payment method. There is a fourth feature similar to the alternative features, but make use of the 'or' principle. On a certain point of the process, you have to choose for either the functionality of feature A or the functionality of feature B.

Since most SaaS applications have an continuous development cycle the updates should also be managed. Moens et al. (Moens et al., 2012) designed a model for developing and managing customizable software. Because multiple users are using the same system it is important that a feature change does not affect the application negatively for (other) users. Therefore features are defined as optional blocks that can be configured. A model where features and their interrelation are defined can be translated in logical statements. The statements are used for designing a correct configuration map and deployment in a runtime environment. This model aims to deliver a high quality of service for customizable SaaS to all different clients. In a later research Moens et al. (Moens et al., 2016) showed that multi-tenancy in SaaS applications can be organised with multiple instances of the software. When there is a separate instance for a tenant client specific requirements can be implemented, without affecting other instances. For managing the customizable SaaS a model is developed where each application is connected with a configuration. The configuration determines which features are included in a certain instance of the application. This model allows two strategies for allocating cloud resources to the applications. Complete instances of a certain configuration can

be deployed on a server (Application-Based Binary), but it is also possible to deploy feature instances on a server (Feature-Based Binary). Besides a hybrid approach is possible where common used application features are compiled in a single instance and additional features are included by using the feature-based binary approach.

An idea for application can be originated as application for one tenant, but might be expanded as multi-tenant application.

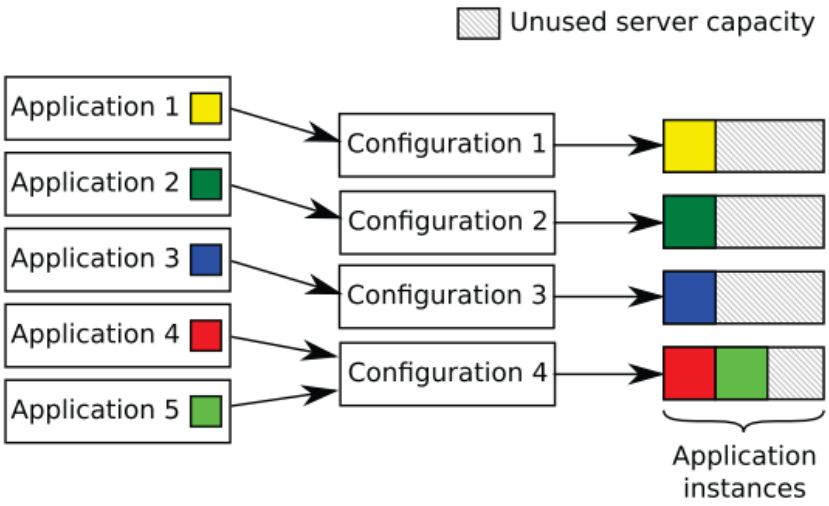

Figure 7: *Application-Based Binary* (Moens et al., 2016)

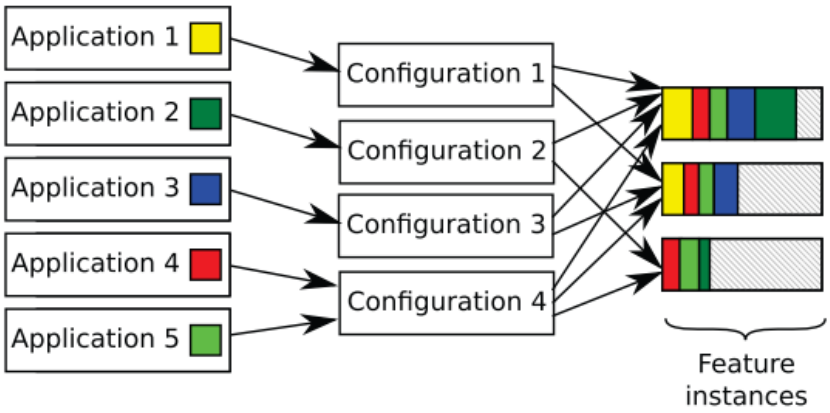

Figure 8: *Feature-Based Bianry* (Moens et al., 2016)

The customizability of the application be can realised by variations. There are two types of variations: compile time variation and runtime variation. Compile time variation is the most flexible type of variation, because a feature can be build by writing custom code. (Moens & De Turck, 2014) The code can be added to one single instance for a certain tenant. Code changes have effect on the whole instance, when a tenant has a specific variation in contrast with other tenants or the original functionality of the application, a separate instance for the tenant can be set up. Runtime variations can be realised by configuration in the application. This type of variation can be configured per tenant or even on user level. For these variations no internal code changes are needed. However, this type of variation is less flexible than compile time variations, for the functions for customizability

have to integrated in the application. Due to the differences of implementation and the size of effect on the application for both variation types, there are two different development strategies.

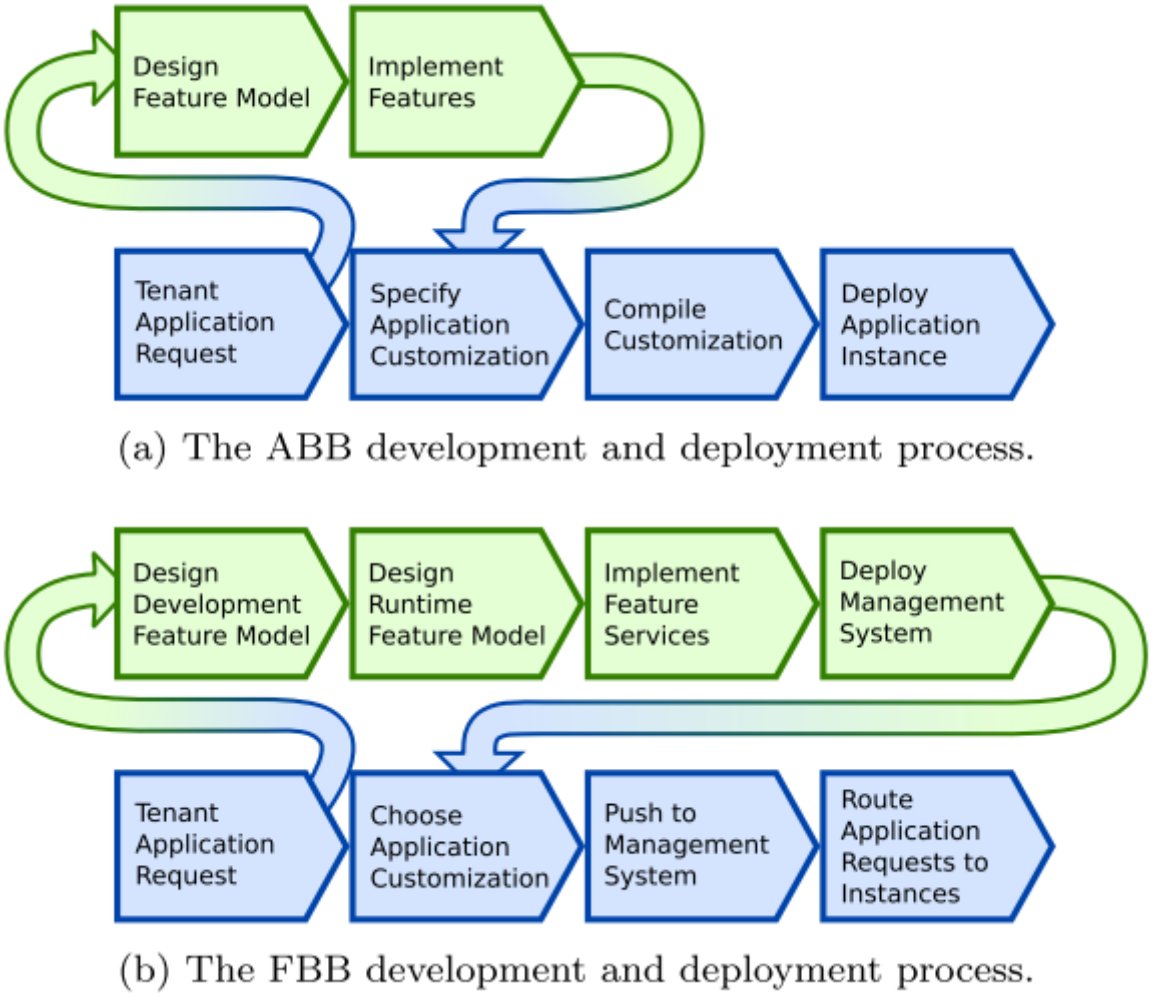

Figure 9: *Development and Deployment Processes for Compile Time Variation* (Hendrik Moens & De Turck, 2014)

4 Analysis Overview

Considering all the found literature there is no method for successful implementation of existing software in the primary business processes of an enterprise. However the found literature covers partially steps that are required during the implementation of customizable SaaS. Mainly the sections 'Change Management', 'Software Development', and 'Data Migration' are interesting and relevant as components of a complete implementation method. Where data migration will be an obvisous step in the implementation method, change management will be less visibile as step, but interwoven throughout the complete process.

Furthermore the customizability of the application is discussed in little detail. Customization can be established on different levels, namely the available services in the application, the SPLs that are realised by the application and service agreements. The results did not cover the customizability by continuous development of the software in combination with co-creation. Besides it might be needed to develop new features in addition to the current state of the software in order to support the business processes of a new client. The proposed methods for achieving customizability are aiming on the design of customizable applications instead of informing about how to handle with existing and connecting new customers on the same SaaS solution.

Literature regarding agile project management is moreover focused on delivering conitinously stable software. Building stable software is important and the techniques that are proposed can be used within agile software development methods, but it does not dwell on implementation methods.

It can be concluded that there is a knowledge gap for the implementation of existing SaaS applications (applications that are already used in production) at the and continuously development of the software. Since there is no complete implementation method for customizable SaaS applications and the need from the business for such a method, brings me to the design of such a method. Though parts of the implementation method can be derived from the found literature, since they can be used within the implementation method.

The way of cooperation between ASP and is not clearly prescribed or highly recommend by the literature. It seems that the cooperation between ASP and depends on a number of factors. involved is partially defined by the chosen software development method, which directly has an effect on the cooperation between ASP and . Besides cooperation is dependent on relation between the parties and the made agreements, mutual commitments and attitude.

5 Implementation Method

This chapter describes the phases of the implementation method. The method provides clear of all steps in the process. It is developed with a hybrid approach of Waterfall and Scrum in mind. The traditional approach is effective for projects with small to no requirement changes, where Scrum is suitable for handling requirement changes (Mahalakshmi & Sundararajan, 2013). However, this implementation method will differ from the standard approaches, for there is already a working software product. The hybrid approach will be applied for adjusting the software where needed and for having a clear implementation plan. The traditional waterfall approach will be used in the start of the implementation method for gathering the required information and for defining the scope of the project. Waterfall is a more predictive software development method than agile methods. Using Waterfall in the first steps of the implementation method helps in having a better insight in impact and planning the to be developed features. In order to be flexible during the implementation, Scrum will be applied as agile software development method. One of the advantages of using Scrum for developing software is that errors are fixed during the development of the software in contrary to Waterfall, where errors are fixed after completing all requirements (Cocco et al., 2011). When it appears that unforeseen requests must be added during the software development method, it is clear that the scope changes, which has an effect on the duration of the implementation phase. Off course unforeseen requests are as well as possible omitted by the “Fit Gap Analysis” and “Define Product” steps in the implementation method. Using a hybrid approach should make the implementation phase more predictable, omitting delays during the implementation for receiving information of other parties and still have the flexibility to test, evaluate and change functionality during the software development.

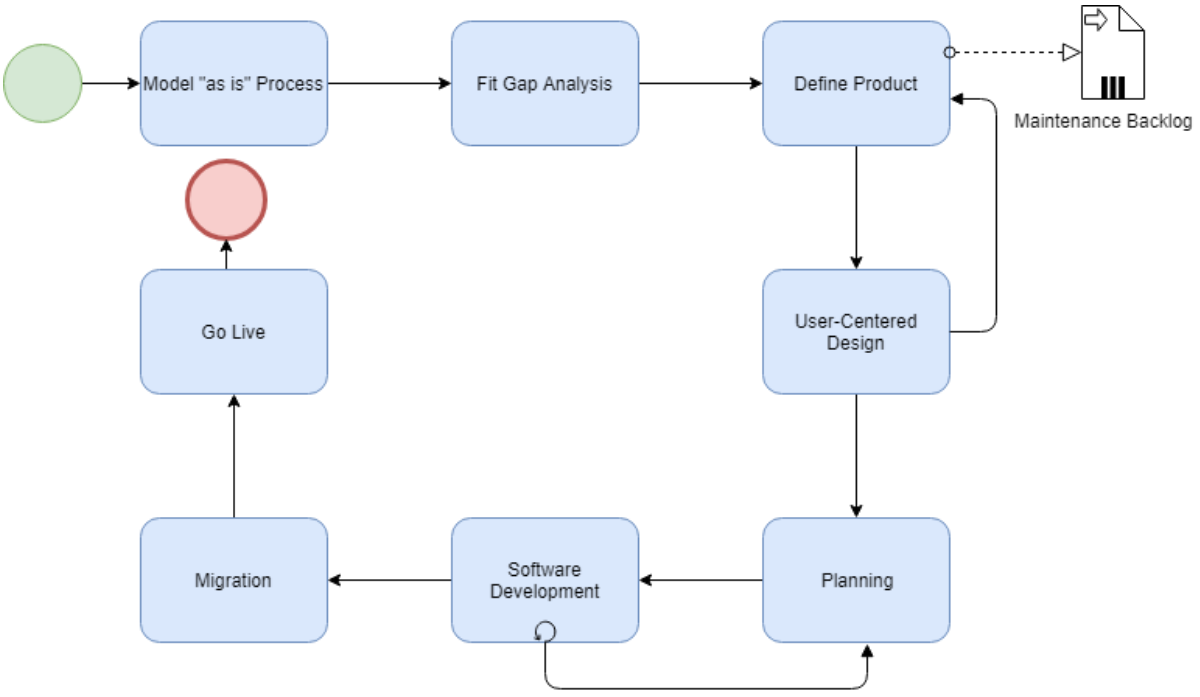

Figure 10: *Implementation Method*

Actors

In the implementation method two main actors are identified. Naturally in an implementation project there will be more actors involved in the project. Still there are two main actors that form the link between customer and ASP. The two main actors are:

- Product owner
- Superuser

In the implementation method, the actor product owner is mentioned often, since there will be one or two dedicated persons for managing the implementation. The product owner is an actor in service of the ASP. The product owner is the link between the ASP and the customer. The customer also has to assign one or more persons to manage the project on the customer side. These persons are often the superusers and must have detailed process knowledge in order to support the implementation of the new SaaS solution. When the customer is mentioned in the implementation method, the required action is often being addressed to the superusers.

5.1 Model “as is” Process

The choice for new software can be derived from the desire to optimize current business processes. IT is seen as one of the great enablers of change in organizations. IT in itself is not able to make the change, but it is supporting in optimizing processes (Davenport, 1993). Business Process Reenginering (BPR) is a method that can be used for optimizing the process. The method has certain phases, whereas the first stage is to model the “as is process”. Requirements for this phase are “a clear understanding of the , market, industry and competitive directions” (Attaran, 2004).

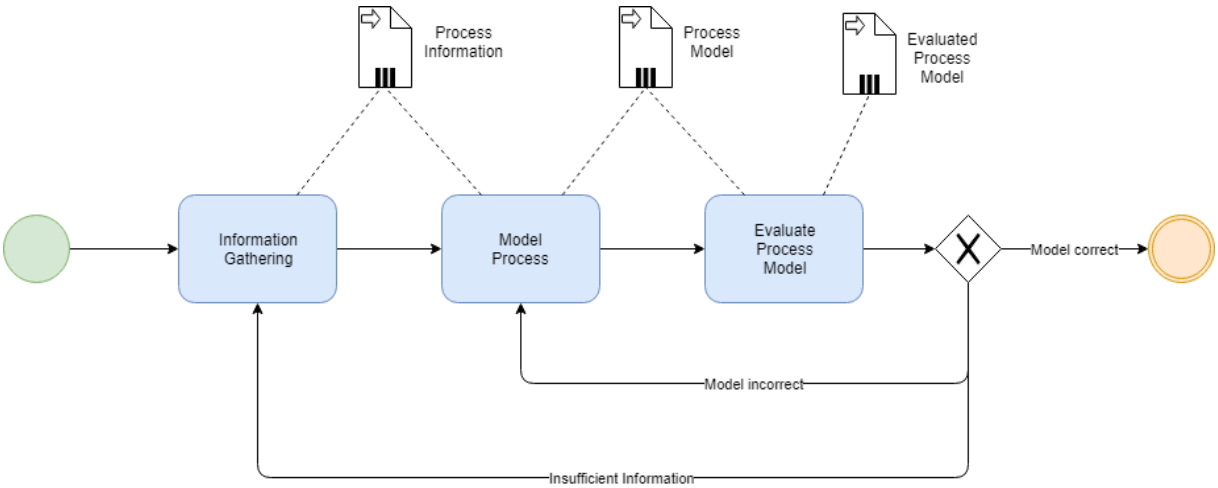

Figure 11: *Model “as is” Process*

5.1.1 Information Gathering

Input

When modelling business processes it is important to gather information as first step. A clear understanding about the processes and the corresponding inputs and outputs of the processes are required for modeling the “as is” process. Information should be gained about the processes that will be concerned during the reengineering. In case of software replacement by SaaS, this will be the processes that are currently performed in the software package that is going to be replaced. Besides all other processes that will covered in the new SaaS solution, but currently don’t have a supporting software solution.

Method

Information will be gathered during a field research. The ASP has to assign a product owner for the implementation project. The should provide availability of their customers. The product owner and customers will examine the current processes. Besides the inputs and outputs of these processes will be gathered.

Output

After gathering the information the following outputs could be delivered:

- Transcripts of the interviews with the customers
- Process documentation of the current process
- Screenshots of the current used system as addition for extra information regarding the transcripts
- Contact details of stakeholders, both internal as external

5.1.2 Model Process

Input

The gathered information is usable for modelling the processes of the . These models help in getting a clear understanding of the current process and possible bottlenecks can be identified. (Ko, 2009) (O'Neill & Sohal, 1999) Besides the product owner gets closely involved with the processes. The communication will also be supported by an understanding of the existing processes (Davenport, 1993).In the end the current process models can be used to check if all processes are supported by the new SaaS solution.

Method

The Business Process Model and Notation (BPMN) (Object Management Group, 2011) is a standard for modelling business processes. It is readable and understandable for both technical and non-technical people. This also improves the common understanding of the current existing processes.

Output

The result of this action are the business process models of the processes that will be supported by the new SaaS solution. As an addition to the process descriptions specification of the inputs and outputs related to process can be delivered.

5.1.3 Evaluate Process Model

Input

The designed process models are the input for this step in the process. A correct and clear design of the process models is important. During the rest of the implementation project, these models can be used as source.

Method

The product owner and the customers will evaluate the business process models. This evaluation has three main goals.

- Check if there are misconceptions in the process models;
- check if all processes have been covered by the models;
- and if the models are correct, confirmation of the common understanding of the processes.

Evaluation of the processes can be done best in a physical session. When a process is unclear, it should be tested in practice. After testing the process it graded as correct or incorrect.

Output

After approving the designed process models, the models and information can be completed for the next step of the implementation method. The models can be shared in team folders and shared with the software development team.

5.2 Fit Gap Analysis

The fit gap analysis is a technique to find the alignment between business processes and technology (Pajk, 2013). The analysis indicates fits of the software for supporting the business processes, but also the gaps. The found gaps have to be closed during the implementation process before the software is usable for the customer.

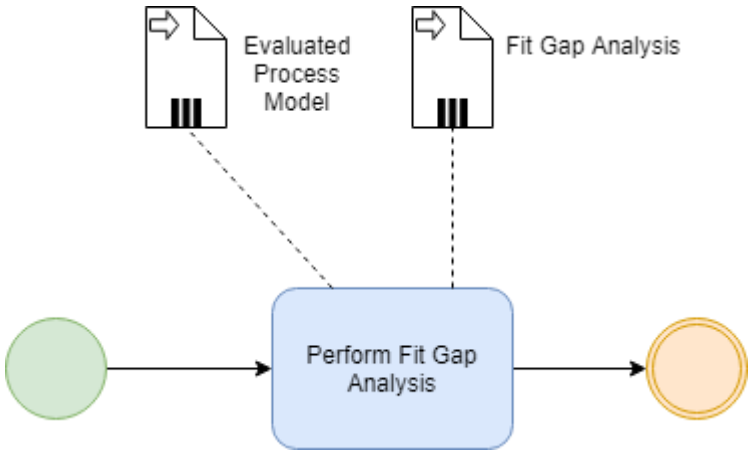

Figure 12: *Fit Gap Analysis*

5.2.1 Perform Fit Gap Analysis

Input

The gained common understanding of the processes to be reengineered and executable processes in the new SaaS solution are the basis of the fit gap analysis. The expertise and knowledge about the to be implemented software is required for matching the chosen software with the designed “as is” process models (Gulledge, 2006). The “as is” process models serve as reference models for the fit gap analysis (Pajk, 2013)

Method

The method and the approach of the fit gap analysis differs from standard methods. Usually business processes are reengineered to achieve certain optimization goals. After reengineering the business process a fit gap analysis is applied to check and compare information systems on compliance with designed processes. (Gulledge, 2006; Pajk, 2013) The difference in order steps originates from business model of customizable SaaS. Before selecting a software provider, the often performs an analysis for selecting a provider. This other approach fits in this implementation method, for it is designed from the point of view of an ASP. Pajk (Pajk, 2013) presents two types of fit gap analyses, the high level and detailed fit gap analysis. The high level fit gap analysis is mostly used for the selection procedure of a software solution. The detailed fit gap analysis is used during the implementation of a selected software solution. The detailed analysis will be used in this step of the method. The fit gap analysis can be performed by testing the evaluated process models in the new SaaS solution.

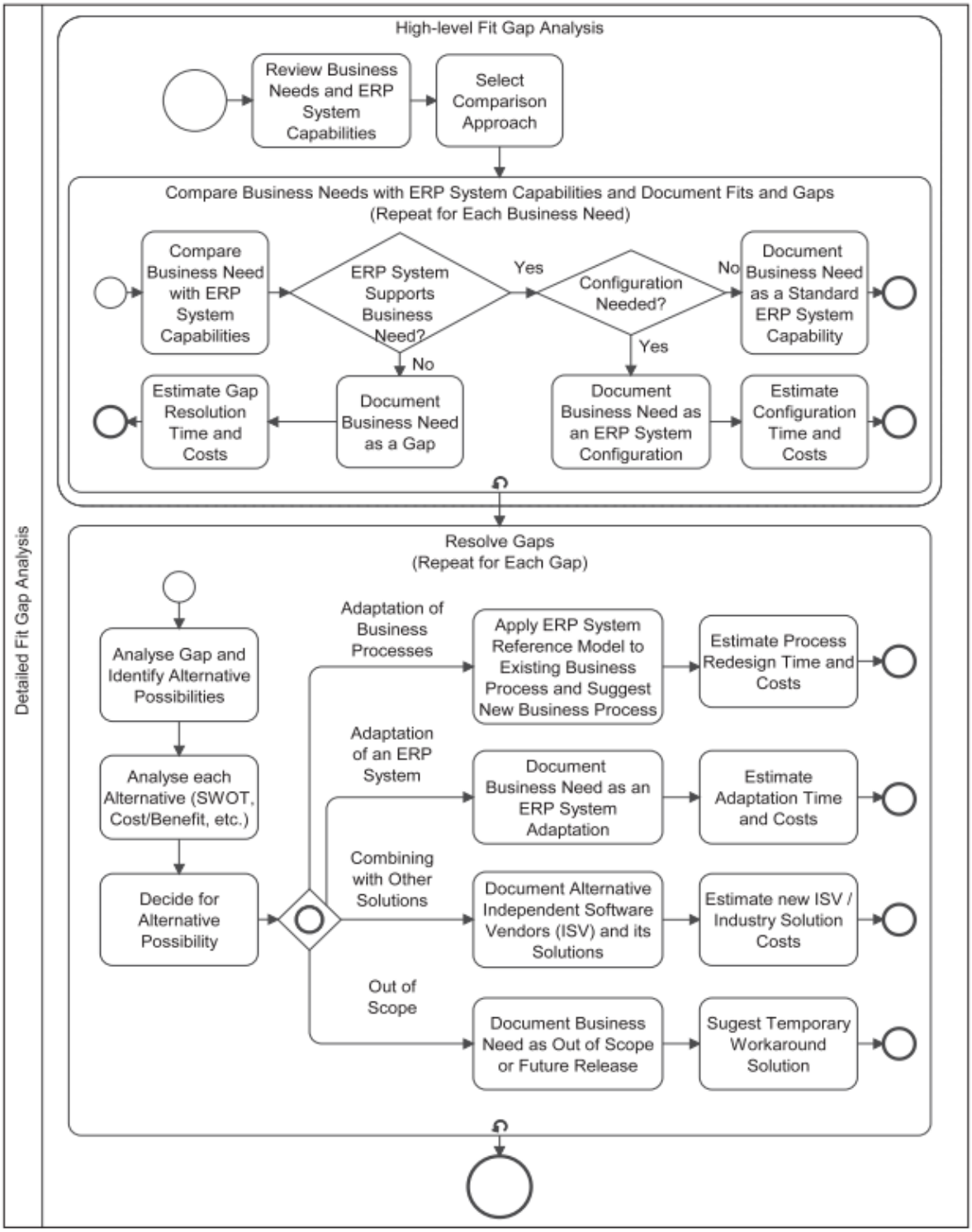

Figure 13: *Detailed Fit Gap Analysis Process* (Pajk, 2013)

Output

The gaps found during the analysis should be documented. All the found gaps together form the fit gap analysis. Since the SaaS solution will replace a current system and it has an effect on the primary business processes, most of the gaps will be resolved by an adaption of the system.

5.3 Define Product

In the fit gap analysis are the found gaps between current business processes and the SaaS solution described. In addition to solutions for closing the gaps, the might requests for additional features. The requested features are often an extra stimulant for changing from software vendor. The found gaps and requested features can be translated to user stories. The user stories end up on the backlog. After defining the user stories, the minimum viable product (MVP) can be defined. The MVP is a subset of the defined user stories, which must be completed before going live. The user stories that are not in the subset of the MVP can be developed for this after going live. Both user stories and MVP must be evaluated by the and the product owner, so that scope is clear before starting the development of new software components. This approach fits in the hybrid Waterfall-Scrum strategy which is used in this implementation method.

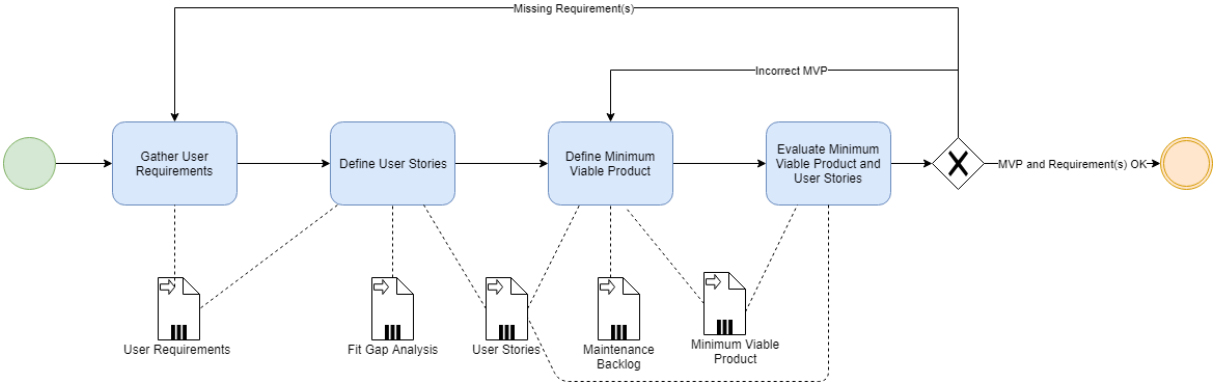

Figure 14: *Define Product*

5.3.1 Gather User Requirements

Input

In addition to the functionality offered by the new SaaS solution and then found gaps that must closed, the customer can aslo request other features. Since there is already an established business process and in most cases a working software product, the customer can identify where these features can add value in the process. Detailed information about the requested functionality and the business rules to which the feature has to apply are necessary as guidance in the software development step of the method. Besides it is critical in omitting mismatches between the processes and the requirements of the supporting software (Boehm & Turner, 2005).

Method

Lauesen (Lauesen, 2002) developed a method for classifying requirements into four different levels:

- Goal-level
- Domain-level
- Product-level
- Design-level

Classifying the requirements is helpful in defining user stories and in a further stage for creating the functional designs, because the level of impact of the requirement is known. The goal-level requirements describe which goal or KPI must be met when making use of the application. Domain-level requirements describe which features must be included by describing the tasks to be supported. The product-level requirements state the required functions of the applications. Finally, the design-level requirements describe which design requirements there are.

Output

At the end of this a task user requirements should be described and categorized in the four requirement levels. These requirements should clarify and describe feature functionality that has to be built. Goal-level requirements can be seen more as an overall goal rather than a hard requirement. Most of the requirements that will be gathered will be within the domain-level, product-level and design-level requirement areas.

5.3.2 Define User Stories

Input

The fit gap analysis and the user requirements will be the source for defining the user stories. Insights gained from conversations should provide the product owner with most of all details. The goal is to create an overview of the required functionality as complete as possible. A complete overview of user stories can omit change requests that could be foreseen in advance, during the implementation.

Method

The gathered user requirements and the found gaps from the fit gap analysis can be written down as user stories. During the informative sessions with the customer, user requirements are defined. These requirements are often described from a customer perspective. The requirements are possible feature requests, which should be split up in users tasks. A feature or requirement can generate a set of user stories, which clarifies the functionality of a feature or properties of a requirement in more specific detail. The user stories can be planned on a Scrum board. User stories are manageable programming tasks for the software developers (Karlesky & Voord, 2015). Each user story must contain the following elements (Lucassen, Dalpiaz, Werf, & Brinkkemper, n.d.):

- For *whom* is the feature built
- *What* the user expect from the system
- *Why* the functionality is expected from the system (optional)

Lucassen et al(Lucassen et al., n.d.) defined the following criteria for defining high quality user stories.

Criteria	Description
Syntactic	
- Atomic	A user story expresses a requirement for exactly one feature
- Minimal	A user story contains nothing more than role, means and ends
- Well-formed	A user story includes at least a role and a means
Semantic	
- Conflict-free	A user story should not be inconsistent with any other user story
- Conceptually sound	The means expresses a feature and the ends expresses a rationale, not something else
- Problem-oriented	A user story only specifies the problem, not the solution to it
- Unambiguous	A user story avoids terms or abstractions that may lead to multiple interpretations
Pragmatic	
- Complete	Implementing a set of user stories creates a feature-complete application, no steps are missing
- Explicit dependencies	Link all unavoidable, non-obvious dependencies on user stories
- Full sentence	A user story is a well-formed full sentence
- Independent	The user story is self-contained, avoiding inherent dependencies on other user stories
- Scalable	User stories do not denote too coarse-grained requirements that are difficult to plan and prioritize
- Uniform	All user stories follow roughly the same template
- Unique	Every user story is unique, duplicates are avoided

Figure 15: *Quality User Story Framework*

Output

The deliverables of this tasks are all the user stories retrieved from the fit gap analysis and the user requirements. The user stories can also be grouped by feature/functionality. The priority and dependencies on other user stories can possible give more depth insight, which will usable in the planning task.

5.3.3 Define Minimum Viable Product

Whereas a minimum viable product (MVP) is used much, in defining the requirements for a prototype, in order to receive valuable feedback from the early adopters (Lenarduzzi & Taibi, 2016). The minimum viable product will be used in this method to define the scope of the software implementation project before the 'go live' moment.

Input

The input for defining the MVP can be retrieved from the user stories that were defined upon the fit gap analysis and user requirements in the previous steps. Identified gaps and additional requirements from the client are used as input for defining the MVP. The product owner and client can discuss the mandatory features during the user requirements gathering. For each stated requirement it is important to have clear, how essential the requirement is for making improvements to the current process. Keeping in mind that BPR has as goal to improve current processes (Davenport, 1993), which will be supported by the SaaS solution you have as ASP.

Method

In traditional software development cycles the scope of the project is equal to defining the requirements. In agile approaches the scope is more project focused rather than product focused (Boehm & Turner, 2005). This enables the software provider to assign a subset of the user stories to the project scope. The user stories that are out of scope are automatically assigned to the backlog for implementation after go live.

Output

The MVP shows an overview of the user stories that will be developed before the 'go live' moment. In order to get a clear overview the user stories should be grouped per functionality and in addition a short description of the complete functionality can be added. The MVP should also contain a section describing which user stories are defined but are out of the scope for the MVP. This section informs the customer about the cognizance of the requested functionality.

5.3.4 Evaluate Minimum Viable Product and User Stories

Input

The defined MVP is the subject of discussion in this task. In the lead to here there should have spoken between the and ASP for the establishment of the MVP and the user stories that it contains. A final check and agreement from the is sufficient to proceed to the impact phase.

Method

The product owner has to evaluate the MVP together with the customer. The MVP has to be evaluated on the following points:

- Completeness of the user stories
- Defined scope of the MVP

It is best practice to evaluate the MVP in a physical session with the . Any notes about the MVP can directly processed in a revised version of the MVP, when the number of adjustments is low.

Output

An agreed MVP will be the deliverable. The MVP reflects the scope of the project and can also contain commitments from the ASP to the for feature development after going live. The evaluated MVP should be accepted by the directors or main responsible person of both the ASP and in order to omit future discussion about the evaluated MVP.

5.4 User-Centered Design

Based upon the defined user stories and the MVP designs for the to be implemented features can be made. These designs will provide information to project team and is usable as reference. The scope of the project and requested features and their appliances will be more clear to the team. Well developed designs ensure that development of new features can continue even without extensive contact between the and the project team. As this method is used for existing applications new features should be integrated with the current system. Therefore the won't be involved as much as in complete agile methods. Besides all s share the same codebase and therefore design decisions should be made by the project team or product owner and inspired by the customer.

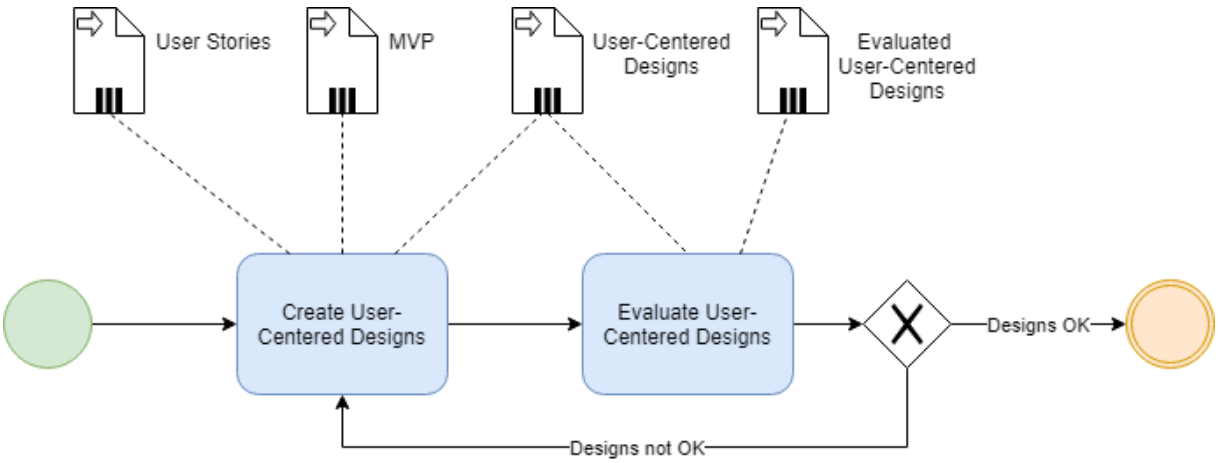

Figure 16: *User-Centered Design*

5.4.1 Create User-Centered Designs

Input

Creating designs is a component of the traditional development strategies, where the designs are created before the development phase. In Agile methods, the creation of designs is uncommon. Agile methods like Scrum have in contrast with traditional software development methods more user participation it the project (Chamberlain, Sharp, & Maiden, 2006). User-centered design (UCD) is a technique where the user takes a central role in the design. The designs are described from a user perspective, explaining the functionality from a user viewpoint. The context and user stories are defined and will be used together with the MVP for developing the designs. Putting the user in the center.

Method

The UCD will be developed on top of a set of user stories that belong to one feature. The UCD has to contain the following elements:

- Goal of the feature
- Feature description
- Usage description
- Prototype
- Corresponding user stories

The goal of the feature will provide an explanation why the feature is requested and can give extra context information for the software developer. The basic principles are explained in the feature description. When the user or product owner has an idea for the working of the feature, it can be described in the usage description. A prototype or sketch of the features will be valuable for the frontend designer to implement the features in the right visualization. At last the corresponding user stories help to capture the overall scope of the feature and it can be used for keeping track of the user stories on the backlog.

Output

The designs delivered contain information about the features that have to developed and the corresponding user stories. These artifacts will be used as guidance by the software developers. It should provide them an insight in the perspective, requirements and how the feature will used. Costly revisions of the built software should be omitted as much as possible by using these designs as guide, for they are user-centered developed. Besides the customer is able to verify the design before all user stories are implemented. Design prototyping is faster than development prototyping (Chamberlain et al., 2006) and can be used to omit unnecessary design revisions.

5.4.2 Evaluate User-Centererd Designs

Input

The created UCDs will server as input for the evaluation with the customer. Evaluating the designs with the customer omits misunderstandings about the features that are going to be built. Besides as ASP you also prevent a waste of resources by evaluating and validating the design before starting the development. Evaluation also contributes to the acceptance of the feature that will be built based on the design, since the design is approved by the customer.

Method

Evaluation of the deisgn should be done by the product owner as representative of the ASP and the superusers of the customer. The superusers should validate the design against the requirements from the business. After the validation of the design the superusers can approve or decline the concept. When the concept iis declined, adaptions should be made to design or even a complete new design can be created based on the gained insights.

Output

The evaluated UCDs are the output of this step. When the evaluated UCDs differ from the original designs, it also impacts the defined product. Therefore it it could be that based on the evaluated design it is needed to describe new users stories and add these to the MVP as described in the phase ‘Define Product’. The evaluated design will be the guide in the ‘Software Development’ phase for the developers.

5.5 Planning

Planning is the process where the scope of the project is expressed in time and required resources. An accurate planning is important for scheduling budget, resources, time and costs (Nasir & Study, 2006). In the business models for SaaS applications, mainly the required resources and time and internal budget play a role, since the is being charged for the use of application instead of charging the directly for the development. Having an inaccurate planning estimation may have internal and external effects.

An inaccurate planning can request more internal budget, resources or time, which can affect the planning for other projects. On the other hand, the will very likely have acted on the estimated planning within in their organization. Personnel capacity may be reserved for implementation on both the customer side as the ASP side. Besides IT contracts of other software suppliers may already have been ended.

For making a planning, several steps are must be taken. It is required to make an estimation of the scope of the project. After estimating the scope of the project, resources should be allocated to the project.

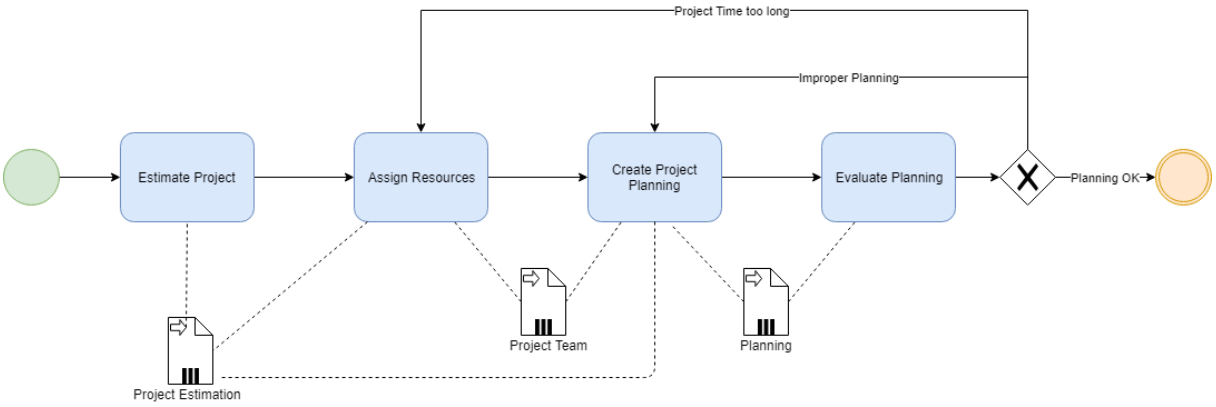

Figure 17: *Planning*

5.5.1 Estimate Project

There are three methodologies for making an estimation (Nasir & Study, 2006):

- Analogy Method
- Top Down Method
- Bottom Up Method

The analogy method requires historical data of similar projects. By comparing the current project with completed projects, it is possible to make an estimation for the planning. The top down method is focused on the characteristics of the project and is more an abstract method for estimating the project. The bottom up method performs an estimation per component (Nasir & Study, 2006). Since the project has a backlog with user stories, the bottom up method can be applied for making an estimation per user story.

Input

The project will be estimated by story points. The project backlog is the resource that contains all items that have to be completed before the goes live with the software.

Method

Estimation backlog items will be done by assigning story points. Assigning story points is done during a team session while playing planning poker. The idea behind planning poker is that each team member makes an estimation of the required effort by playing a card with a number of points. Each team member has the same set of cards. All team members shown their card at the same moment, in order to avoid team members influencing each other. The team members that have a higher or lower estimate than the average team members are allowed to discuss their estimation until a consensus has been reached in team. This process must be repeated for each item.

Output

When each backlog item is reviewed, it is known how much it will probably costs to complete the project. The number of estimated story will also play a part in assigning resources to the project.

5.5.2 Assign Resources

The projects for customizable SaaS solutions differs from traditional project in the maintenance phase. Were in traditional projects a request for change in the maintenance phase can be seen as a new project with it's own scope, customizable SaaS is often treated as software on demand as maintenance is included in the standard fee. Therefore there should be two teams, one for maintenance and one for new implementations, when the number of s is growing. Having a dedicated team for new implementations ensures that a certain allocation of resources is only working on implementation, which will result in a more reliable planning.

Input

For making the right distribution of resources, a number of items must be inventoried. The backlog items that are included in the MVP must be completed before going live. An estimate for these items has been made in the previous step. A first suggestion or goal for going live may already been given by the and can be included the process of allocating resources. It is also possible that there are multiple projects simultaneously for new implementations, which have to be taken into account.

Method

Each developer has to be evaluated on its velocity expressed in story points per sprint. Also the ratio between the maintenance team and the implementation team has to be defined. Based on the ratio and total velocity , a composition for the two teams can be established. It is possible to exchange team members between the two teams during the implementation, as long as the ratio will be retained. When the ratio changes, this will have an impact on the planning.

Output

The composition of implementation team and its velocity is defined and can be used creating a planning.

5.5.3 Create Project Planning

Input

The backlog items that are included in the MVP and their story points and velocity of the assigned team are required to make an effective planning. The planning for other new implementations and their backlog items also have to be considered.

Method

The total estimated effort needed for the project and the project team are known. The follow-up is the creation of an accurate planning. First the order in which the items must be completed should be determined so that dependent items are built in a logical order. Based on the total number of story points and the team's velocity, the total required sprints can be created. The backlog items must be planned in logical order from the first sprint to the last. When features are independent from each other, then the corresponding features can be built in parallel.

Output

Finally a planning on items to be completed on sprint level will be delivered. This detailed planning can also be summarized in a brief planning, that can be used for communication with the .

5.5.4 Evaluate Planning

Input

The created planning is an estimation from the team based upon the open backlog items and estimated effort for each effort. However, the might have remarks or disagrees with the planning. Therefore it is important to have a meeting with the to discuss the planning.

Method

The final approval of the planning should be done in cooperation with the . The product owner and customer have to evaluate the planning. The evaluation contributes to the involvement, expectation and the common understanding. A close relation and common understanding keeps the communicative barrier low. The evaluation might request for a revised version of the planning.

Output

The evaluaion will show whether the planning is agreed or not. Since the MVP was accepted in an earlier stage, no other requests should be added by the . The might possibly disagree with the project time or the order in which features are released. When concluding that the project time is too long, it is needed to move back to the task 'Assign Resources'. If the planning is improper, the planning of items in the sprints have to be revised. In case the planning is agreed by the , the planning could be endorsed by the managing directors or project supervisors of both the ASP and to show their mutual commitment.

5.6 Software Development

The implementation method uses a hybrid approach of traditional and agile software development. The previous steps in the implementation method were predominantly using the traditional approach. In order to have flexibility during the actual software development phase, an agile approach has to be applied in this phase.

There is a variety of agile software development methods and frameworks. It is important to consider how the previous steps in this implementation methodology will fit with the chosen software development method. Since the backlog and planning already has been made up, the software development should fit this approach. Besides the software development should also be able to track delays and possible wrong assumptions in the planning in order to tackle problems as soon as possible.

A software development method that fits with previous defined phases in this implementation method is Scrum. The created backlog and effort estimation using planning poker are parts of Scrum.

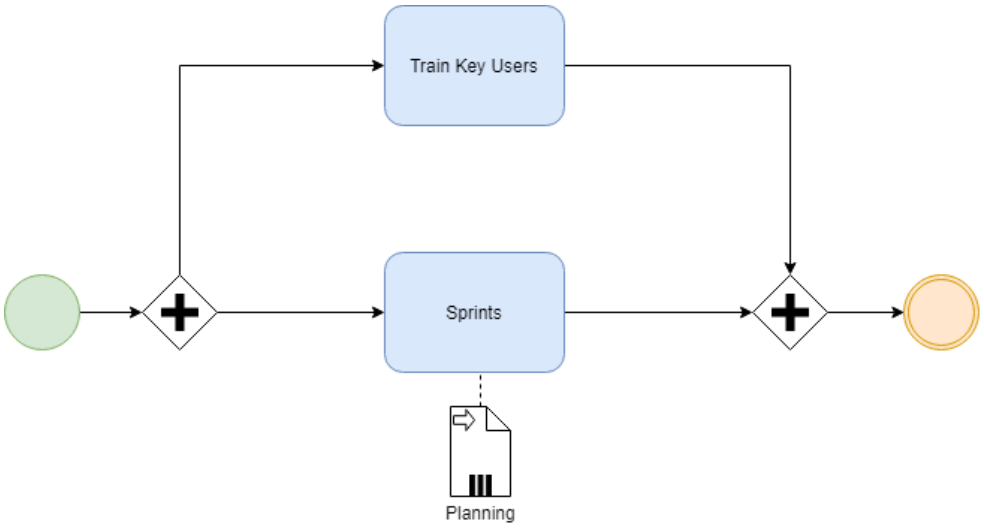

Figure 18: *Software Development*

5.6.1 Sprints

Input

The established planning will be used during the software development phase. The step 'Sprints' is a repeating cycle until all sprints are finished. The standard input for a sprint is the established 'Sprint Backlog'. Since one of the aims is having an accurate planning, a first planning for all sprints was made in the planning phase.

Method

Scrum is a method for organizing a project in an agile way. There is a wide variety of Scrum guides, explantions and and reviews of use cases. Being agile means that flexibility is brought into the process in comparison to the waterfall approach. Therefore no strict standard format has to be applied for Scrum. However, there are basic principles that characteristic Scrum (Deemer, Benefield, Larman, & Vodde, 2010). The main practices to keep control on your project are:

- Sprint Planning
- Daily Standup
- Sprint Review

In the Sprint planning a subset is taken from the product backlog that is considered a reasonable amount of effort for the next Sprint. In this method, the Sprint backlogs are set-up in advance, since an estimate effort is done before the start of the software development. However, when estimation is not accurate, the scope of the planned sprints might change.

The daily standup brings the team together and raises possible problems in an early stage. Besides it is a daily review on the progress and productivity of the team.

At the end of each sprint, a sprint review should be held. In the sprint review, the team discusses the work done in the last sprint. The sprint burndown chart shows the work done in story points and let the reflect on their productivity.

Output

At the end of each sprint iteration the output of the sprint review can be used for adjusting the team's velocity and adjust the planning the new established team velocity. Besides each sprint

delivers new functions for the software, which can be tested by the . When all backlog items are finished, the development phase can be ended.

5.6.2 Training

Input

The training of the customer's superuser(s) is done parallel with the Sprints. Usually training is performed after developing the product, but since there is already a working version of the software that is used by other customers, the superuser can already master the functions and tools offered by the current state of the software. Besides the requested features that were found in the fit gap analysis and are included in the MVP, are implemented in the required sprint cycli, which means that the software will provide new functionalities to the superuser after each sprint.

Method

During the software development phase the product owner regularly meets the superuser(s) for a training session. In the training session the product owner demonstrates the new functions developed in the last sprint(s). After the demonstration the superuser should test the functionality under supervision of the product owner. Possoble feedback can directly evaluated by the product owner.

Output

After training the superuser must be able to train other users withing the customer's company. The superuser will be the first service line for internal questions regarding the new SaaS application. During the training session, feedback can be given about the developed functions. Possible requests for changes might arise during these sessions. Changes in itself are not a problem, but it has an impact on the planning. According to Karlesky and Voord (Karlesky & Voord, 2015), changes should be managed instead of being avoided. Thus when a change is required, it can be accepted, but the consequence is that the scope of the planned sprints will change and thus a later go live moment.

5.7 Migration

Migration is the transfer of data from the the application that is going be replaced to the new SaaS solution. In order to get a higher user acceptance, ensure continuity and prevent overhead, correct migrations are necessary. Since the (crucial) gaps that possibly were found between the current system and new SaaS solution, are closed, the new SaaS solution has function parity. There it should be possible to migrate parts of the data collected in the current system.

"In summary, migration to SaaS requires to consider the specific migration strategy according to legacy system and existing SaaS. If existing SaaS has the same business functionality of legacy system, users can replace legacy system by SaaS." (Zhao & Zhou, 2014)

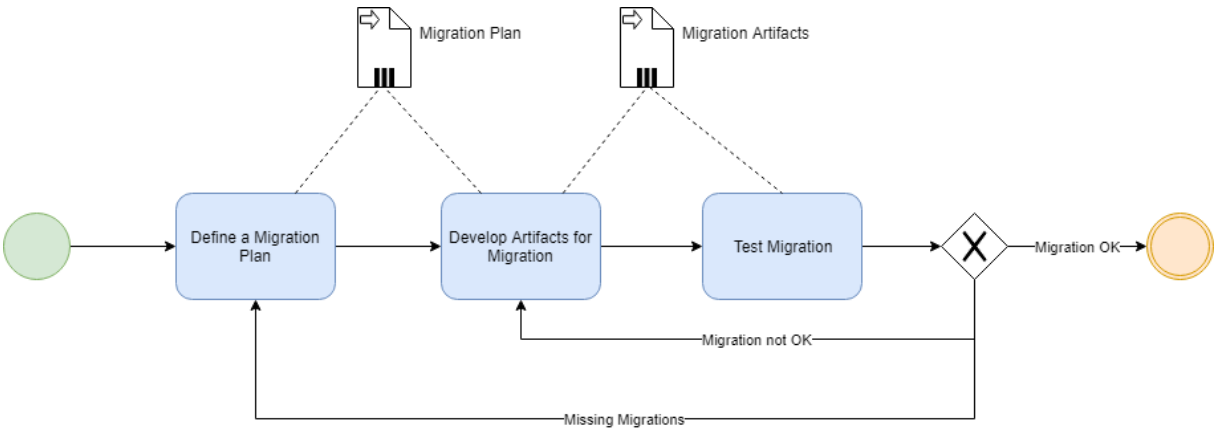

Figure 19: *Migration*

5.7.1 Define a Migration Plan

Input

A migration helps in the processes for tracking down what data should be migration and in determining the right timepath development and testing the migrations. Knowledge about the functionality of the current system and possibilities for data output or database design are necessary for defining a migration plan. In case the current processes are covered by worksheets, an overview of the available data will be used as starting point.

Method

For defining the migration plan it is recommended to have a close cooperation between the customer and product owner. The product owner and customer have to do an assessment of the possibilities regarding the masterdata of the new SaaS solution. Besides the possible exports or data outputs of the legacy system have to be evaluated. Possibly a data engineer or system administrator of the legacy system can also provide information about the available data in the legacy system. A selection for required and valuable data transmission should be made. After selecting the data for migration a plan can be set-up. Since both product owner and customer are involved in the task of defining the plan for migration, both and ASP are aware of what will be migrated at go live.

Output

The data selection for migration and example dumps are part of the migration plan. These data dumps will be used by the ASP for analyzing the properties of the data and for developing the data migration artifacts. Based upon the selected data and the expert knowledge of the product owner, an effort estimate can be made for the required migrations. The ASP should make migrating/importing data from other systems standard functionality. Most customers are likely to have similar migration requests, since they all fit in the same application.

5.7.2 Develop Artifacts for Migration

Input

The migration plan is the starting point for the development of artifacts. When the data dumps and corresponding descriptions does not contain sufficient information for developing the migration artifacts, they should be complemented by the product owner (or customer).

Method

The developers should analyze the data dumps together with the product owner in order to clarify all details. After clarification of the dumps, mapping models have to be developed as migration artifact. The dumps may contain dependent models which rely certain information in the master data of the legacy system. Therefore developing migration artifacts for the master data or independent data should be used as starting point.

Output

The mapping models or data converted in a readable format are the deliverables for this step. It should be possible to import the selected data into the accepation environment of the new SaaS solution.

5.7.3 Test Migration

Input

The developed artificats have to be tested before the migration can be used in the go live step. The expert knowledge of the product owner will be used for evaluating the artifacts. Besides the customer also has to the evaluate the developed artifacts and has to approve or decline the artifacts.

Method

The migration artificats have to be used to import the data in the acceptation environment. The product owner will inspect the imported data first. Possible modifications or corrections can be made before letting the migrations reviewing by the customer. After acceptation of the developed artificats by the product owner, the artifacts can be evaluated by the customer. Reviewing the artifacts by the customer can be done best in a meeting or demonstration by the product owner.

Output

Testing the migrations can have three different outcomes. The customer has to accept or reject the developed migration artifacts after testing. When the artifacts are rejected they miss either migrations or the migrations contain errors. If there are missing migrations, they should be added in the migration plan and proceed the process from the step ‘Define a migration plan’. In case the migrations contain errors, the concerning artifcats should be corrected. When the migration test is successful, everything is ready for ‘Go Live’.

5.8 Go Live

The Go-live moment is the last phase in the implementation method. In this phase the starts using the system in production. At go-live there is transmission of data between the system that is replaced and the new SaaS solution. Also the implementation project is finalized and the open backlog items will be transferred to the maintenance backlog, as this customer shifts from the implementation process to the maintenance process.

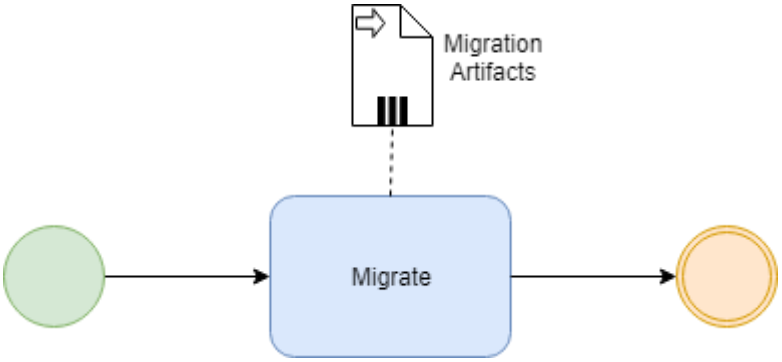

Figure 20: *Go Live*

5.8.1 Migrate

Input

In the migration phase, a plan for migration and the corresponding artificats for migrating were developed. At the moment of go live a new data dump might be provided by the in order to have to most recent data migrated. The developed artifacts should it made easy to import the data dump. Besides the possible renewed data dump, a central kickoff meeting is helpful for creating team spirit within the company.

Method

The developed migration artifcats will be used for migration. When needed a fresh data dump has to be made from the legacy, whereafter no new entries or actions should be entered into the legacy system. The new data dump can now be imported in the new SaaS solution with the help of the migration artifacts. It is recommend to have backup accessable of the data from the legacy system. This is especially a requirement when not all data is migrated from the legacy system to the new SaaS solution. After importing the data, it has to be verified on correctness and completeness by the customer. As the migration artifactions were approved, the migratrion itself should go well.

Output

The migration ensures that the new SaaS solution contains an up to date dataset of the master data and orders from the legacy system,whereby the endusers should be able to use the system without having to insert unnecessary master data manually. The new SaaS solution is now ready for use and the customer shifts to the maintenance phase.

6 Validation

This chapter describes the validation of the designed implementation method in chapter 5. The selected method for validation is shown in section 6.1. In order to introduce the context of the experts, section 6.2 outlines the expert experiences towards implementation projects of customizable SaaS solutions. The answers given to the questions in Appendix A are elaborated in section 6.3. Finally updates to the designed implementation method are described in section 6.4.

6.1 Validation Method

The implementation method is designed based upon field experiences and literature. In order to determine the value of the designed method, validation is done by using expert opnions. The type of validation is a qualitative validation since the method is validated by experts in the field of customizable SaaS implementations. The validation is done by having semi-structured interviews with the experts. This type of interview supports in answering the main questions for validating the fulfillment of the research goal by the method. Besides the experts are able to add other insights to the developed method. The questions for the expert interview can be found in Appendix A. During the interviews, interviewees are allowed to share their insights on the method. These gained insights might also be reviewed in later interviews. The interviews should measure the validity and usability of the designed implementation method.

The questions formulated in Appendix A are based on the measurements in the Technology Acceptance Model (TAM) (Davis, 1989). The perceived usefulness, perceived ease of use and user acceptance should be measured by the interviews.

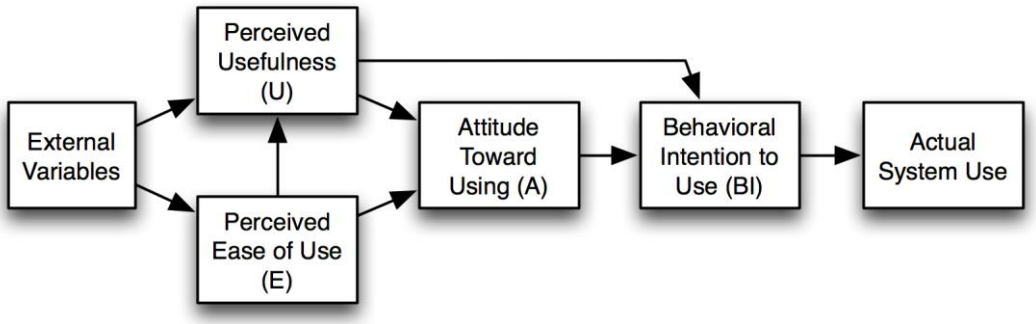

Figure 21: *Technology Acceptance Model* (Davis, 1989)

The model is originally designed to predict the actual system use of an information system. However, the vairiables for predicting the actual system use can also be applied on the designed implementation model. The external variables are determined by the type of product and projects delivered by the experts. The interview questions should measure the perceived usefulness. The perceived ease of use is hard to measure. What is measured about the perceived ease of use is the applicability of the method. When the external variables match with type of implementation, but it is hard to apply the method, the perceived ease of use has not met the right level, which increases the difficulty for implementing the method. The “suitability and completeness of the method” in the second main question of the interview (Appendix A) can be reflected against the perceived usefulness and perceived ease of use. The attitude toward using and behavioral intention to use , are derived from the answers and the extra space for relevant expert opinions during the interview.

For the validation of the method, seven experts in implementations of customizable SaaS were interviewed. The results of the interviews are elaborated below in a textual summary in order to retain the global overview concercing their experiences with implementations and their opnion of the proposed implementation method.

6.2 Expert Experiences

The implementations done by the experts were all in the field of customizable SaaS. The lead time of the implementations had a variation between one month and two years. The implementation time strongly depends on a number of factors:

- Business culture
- Type customer
- Current system
- Resistance from within the customer's company
- Scope changes
- New activities
- Lack of commitment

The business culture defines the division of roles, policies and attitude of the business and it's employees. The difference between a layered and a flat organized business also reflects the interaction between the management who often buys the software and the operation who has to use the software. Also expectations of the new SaaS solution can differ between the management and operation.

The type of customer also plays a role in the implementation time. When there is a new customer of a certain type that will be implemented for the first time will probably take more time, since not all practices and processes will be known to the ASP. Besides customers of the same type typically have the same processes and therefore the functional gap in the software will be probably small to none, when there are already multiple customers of that type using the system.

The business that has chosen the new SaaS solution, might be a brand new company, without any established process, or a business using spreadsheets, or a business with a running system in place. Businesses that have currently a system have usually a longer implementation lead time, due to the request of feature parity. Mainly operational employees are comparing their current system with the new SaaS solution during the implementation method. When their current system is built and maintained in house this can also raise resistance from within the business. Scope changes during the implementation have leaded to delay of the go live for all experts. The reason for scope changes have diverse origins. In a number of cases the processes were not clear for the employees within the business, during the implementation continuously new requests for changes were raised from the diverse departments. Besides there were also business that started with new activitities that were not mentioned at the start of the implementation. The scope changed in order to support the new activities.

Finally a lack of commitment, customer resources or time also has resulted in delays, which negatively effected the implementation lead time. Customer involvement is a requirement for successfully implementing a SaaS solution in the primary business processes.

6.2.1 Current Practices for Implementations

None of the experts currently uses an implementation method for their SaaS solution. However, some practices mentioned in the implementation are used when considered necessary. From the

steps of the designed implementation method, the fit gap analysis, software development, migration and go live are performed by the experts. Not all practices are applied in a structured way for all implementations. As a reason for not using an implementation method, underestimation is mentioned.

6.3 Validation of the Implementation Method

The questions (Appendix A) in the interview had to answer the main question: “Is the designed implementation method suitable and complete for implementations of customizable SaaS?”. All experts confirmed that the implementation method fits for implementations of customizable SaaS solutions. On the other hand, the experts also suggested some minor adaptions or additions to the proposed implementation method. These adaptions and additions are described in the section “Recommendations”.

The expert opnions about the proposed implementation method were positive. They all stated that the proposed implementation method will improve their implementations. Since no of the experts currently used an implementation method, but solely some approaches when considered needed, the projects lack on structure. The structure offered by the implementation method will be beneficial for the ASP and the customer according to the experts. Both ASP and the customer have a clear process to follow and the moments where the customer is involved in the project are defined and clear at the start of the project. The moments of involvement will be the same in each project for each customer. However, the quality of involvement of the customer before and during the agreed moments of contact, heavily depends on the effort the customer takes.

The steps “Define Product”, “User-Centered Design” and “Planning” were mentioned as most helpful steps in improving implementations. These steps will help the ASP in making a better estimation of the project size. Having a dedicated team on new implementations improves the monitoring of the project progress and enables to act early when there is a deviation in the planning. Due to the defined product, scope changes are minimized and when there is a customer request, it can be recognized as scope change. Hence, the scope change has impact on the planning. Involving the customer during the whole implementation process, but most of all in the design phase, will have a positive effect on the acceptance of the new SaaS solution.

Having an implementation method will not only help to manage the project, but will probably also establish confidence in the ASP from customer perspective.

6.3.1 Expert Recommendations

After reviewing the designed implementation method the experts also recommended to expand or adapt the implementation on certain points. The recommendations given by an expert were also validated by experts that were not interviewed yet for validation.The recommendations mentioned are described below:

- Risk analysis
- Stakeholder analysis
- Customer commitment
- User migration
- Parallel steps

Risk Analysis

The designed implementation method helps in having a clear and structured process for the customer and ASP, but does not guarantee the success of an implementation, because the success of an implementation depends on more than only a structured method. The risk analysis can be performed on the steps that are defined in the method and on customer type. Certain can be identified on process step level, these risks can be defined once per SaaS product. The risks that are customer dependent should then be estimated at the start of the product. The risk analysis based on customer type is correlated with the stakeholder analysis, for the commitment and presence of the right stakeholders will probably the greatest risk. One of the experts also added that a risk analysis has not always a positive effect on the implementations, since it can create needless fear for the implementation from a customer perspective. When a lack of commitment is the greatest risk of the implementation project, a commitment agreement instead of a risk analysis can be sufficient.

Stakeholder Analysis

A stakeholder analysis has been suggested by the experts to be done at start of the project. The stakeholder analysis also can help in indentifying the role of the stakeholders in the operational process and their role in the implementation process. The required commitment per stakeholder can then be defined at start of the process, which clearifies the made agreements about the required commitment from the stakeholder at the start of the implementation.

Customer Commitment

Customer commitment is crucial in implementation projects. However, customer commitment is more an agreement between the customer and the ASP, then a step of a process. As mentioned in the risk analysis, concretize the importance of the customer commitment should be done or mentioned at the start of a new implementation.

User Migration

In the implementation method there is a phase dedicated to Migration. In the migration phase a migration plan is made, the artifacts for migration are developed and tested. The focus in the migration phase is on data migration. Besides data migration there is also user migration. User migration is advised to be included in the implementation method. Not only key users have to be trained, but all employees that will start using the new SaaS solution after go live. Therefore user migration should run in parallel with data migration.

Parallel Steps

The high-level process is shown as a sequential process with two feedback loops. One of the experts mentioned that it should be possible to do steps in parallel. The phases “Software Development” and “Migration” can be done in parallel in the practice. The plan for migration and the development of the artifacts can be done when they depend on functionality that is already made. The implementation method should not limit on the flexibility of doing phases in parallel wen possible. Though, showing the method as sequential process, shows the logical order of actions to the customer.

6.4 Updates to the Implementation Method

During the interviews recommendations were given for updating the implementation method. The suggestions for updates do not affect the procedure of the implementation method, but visualize implications that were already made in the textual explanation. Only in the the “User-Centered

Design" I added an extra step for evaluation with the customer on recommendation of multiple experts. The following updates are implemented in the model:

- The step "Define Product" has an explicit output "Maintenance Backlog". The "Maintenance Backlog" contains all user stories that were not recorded in the MVP. The user stories in the "Maintenance Backlog" are assigned to the development team responsible for maintenance.
- The 'User-Centered Design' phase is extend with an evaluation of the UCDs. According to the experts the validation of the designs contributes to acceptance of the designed feature. Besides misunderstandings or possible shortcomings of the design can be identified by the customer before the actual implementation of the feature.
- A loop has been added in from "User-Centered Designed" to "Define Product". Since the experts suggested that it is better to validate the design with the customer and update the MVP before starting the "Planning" step, it is obvious that feedback from the customer might add new user stories to the MVP before the start of the "Planning" and "Software Development" steps.
- A loop has been suggested from "Software Development" to "Planning". During the software development phase Scrum is applied as agile technique. At the end of each sprint there is a sprint review in which the deliverables are reflected against the planning. Whereas it is normal to update the planning each new sprint, it has been suggested by an expert to make this process more visual and explicit by adding a loop for returning to the planning phase. This also enables the user of this method to change the assigned resources during the project.

7 Conclusions

This chapter describes the conclusions of this research. The answers given in this research about the research are summarized. The findings and the deliverable of this research is also presented. The deliverable to the stated research goal is also presented in the conclusion.

The research goal is:

Develop and validate a method for implementation of SaaS solutions in the primary business processes.

In order gather information for developing and validating the design method research questions were defined. The research questions and the answers to these questions can be found in section 7.1. The contribution of this research is presented in section 7.2. Limitations of this research are discussed in section 7.3 Future work related to this research is suggested in section 7.4. Finally the recommendations for using the developed method are given in section 7.5

7.1 Research Questions

In order to develop a method that suffies the research goal the following research questions are answered in the research:

1. How are SaaS solutions implemented in the primary processes of enterprises?
2. Which steps can be identified in the process of migration to a SaaS solution?
3. How can the designed implementation method be validated?

For answering RQ 1 (Research Question) a SLR is performed. The research design for the SLR is defined in chapter 2 followed by the results of the SLR in chapter 3. The results of the SLR are summarized in the analysis overview. The main conclusion is that there was no complete method for implementing SaaS solutions in the primary processes of enterprises. Also the customizable part of SaaS solutions is focused on building a customizable structure for SaaS solutions. However, some literature results were found relevant and are used as a part in the developed implementation method.

For answering RQ 2 a mix between field experiences and literature has been applied. As the SLR returned some usable and relevant concepts as steps or processes within the implementation method, these concepts have been evaluated for application within the model. Identifying the steps for the implementation method is mostly done on expert and field experiences and supported by relevant literature for more in depth support. The identified steps int the process of migration to a SaaS solution are defined in the developed implementation method and can be found in figure 22. Chapter 5 elaborates the steps and the sub-steps of the implementation method.

RQ 3 aims on the validation of the designed implementation method. There are two relevant options for validating the designed implemention method, namely:

- Expert opnion
- Field research

Due to the constraint of time for this research, there was only sufficient time for validation by expert opinion. For validation of the implementation method, questions (Appendix A) were defined for measuring the effectiveness of the implementation method, using the TAM (Davis, 1989). The expert opnion is a valuable tool for validating the method, due to their field experiences. The experts were able to reflect the implementation method on their projects and provided feedback based on their experiences and vision. The validation of the implementation method is provided in chapter 6.

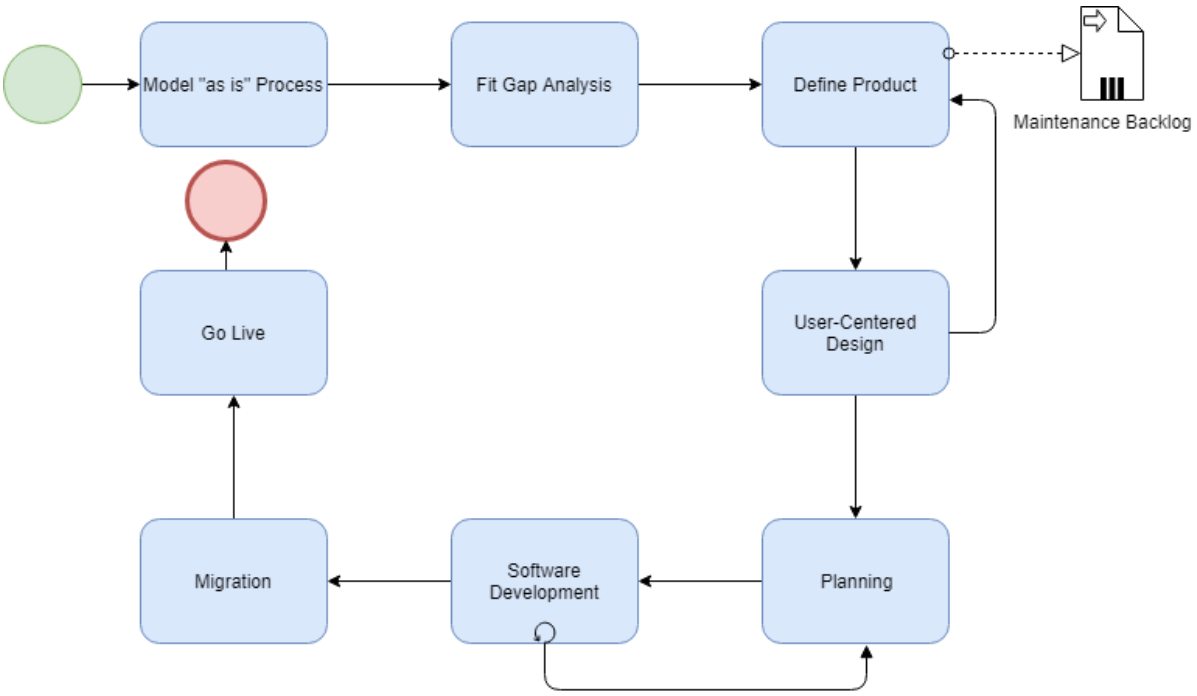

Figure 22: *Implementation Method*

7.2 Contribution

For the implementation of customizable SaaS in the primary process of businesses are no methods available. The structured literature review concluded that there was gap for the design of such an implementation method. Besides none of the experts used a method for their implementation projects. In the SLR some relevant principles are used as a step in the method. Also the experts recognized some of the steps in the designed method. Those steps are already applied sometimes in the projects of the experts. However, a complete method for implementation of customizable SaaS was missing. Based on the found litereature and own experiences an implementation method has been designed. The required steps for implementation are identified in the design of the method and each step is explained by defining the input, method and output of each substep. The validity of the designed implementation method is reviewed by experts in the field of implementation projects for customizable SaaS.

The experts that were interviewed for the validation of the implementation method are positive about the designed implementation method. They all indicated that the implementation method will improve their implementation projects. Also the experts are going to use the method for their implementations. The method provides more insight and clarity about the project for both the ASP and the customer. The commitment of the experts for use of method inidicates their need for an implementation as well as the suitability and completeness of the method. Cofano also stated that they are going to use the method for upcoming projects.

7.3 Limitations

The limitations regarding this research originate in the validation of the designed method. As there was only limited time for perfoming this research, only an expert opnion has been applied for

validation of the method. Despite that the expert opnion is a valuable tool for validating the method, field research could have provided a more constructive validation.

The developed implementation method is designed for a specific type of projects, namely for the implementation of customizable SaaS in the primary business processes. Therefore application of the designed method is limited to to these type of projects. However, the method could be used for similar projects with small deviations. Where needed, the method could be adjusted, so that it suits other type of projects.

7.4 Future Work

Currently the the implementation method has only been validated by expert opnions. For future research it is recommend to validate the method by practicial application. New projects should be supported with the designed implementation method and the outcome of the projects should be compared with projects were no implementation method has been used. When needed the method could be updated after reviewing the method on several projects.

7.5 Recommendations

As Cofano stated that they are going to use the implementation method for upcoming projects I have the following recommendations.

First, inform and structure the organization. Since the implementation method has impact on both consultants/product owners and the development team, all concerned actors should be involved. Besides the development team will be split in two separate teams, one maintenance team and one implementation team.

Secondly, I would recommend to introduce the implementation method to new customers in order to inform them about the process they are going to start. All required actors required for fulfilling this process should also be known at start of the project. Informing the customer also helps them to get more insight in required steps of the project, which improves their monitoring on the project.

Finally, I would recommend to monitor the projects that make use of the developed implementation method in this research. The effectiveness of the method should be evaluated for each step. Possible variations or adjustments on the defined steps should be evaluated by all consultants that are involved with implementation projects in order to have improvement cycle that directly is validated by expert opnions. Customer opnions about this implementation method should also taken into account as valuable information during an evaluation of the implementation method.

8 Bibliography

Agarwal, P. (2011). Continuous SCRUM: agile management of SAAS products. *Proceedings of the 4th India Software Engineering Conference*, 51–60. https://doi.org/http://doi.acm.org/10.1145/1953355.1953362

Ali, H., Moawad, R., & Hosni, A. A. F. (2016). A cloud interoperability broker (CIB) for data migration in SaaS. *Proceedings of 2016 IEEE International Conference on Cloud Computing and Big Data Analysis, ICCCBDA 2016*, 250–256. https://doi.org/10.1109/ICCCBDA.2016.7529566

Attaran, M. (2004). Exploring the relationship between information technology and business process reengineering, *41*, 585–596. https://doi.org/10.1016/S0378-7206(03)00098-3

Aulbach, S., Grust, T., Jacobs, D., Kemper, A., & Rittinger, J. (2008). Multi-tenant databases for software as a service. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data - SIGMOD '08*, 1195. https://doi.org/10.1145/1376616.1376736

Bajighar, M., & Shahzad, F. (2017). Autonomous software development: Sustain market share in constantly changing software product industries. *WMSCI 2017 - 21st World Multi-Conference on Systemics, Cybernetics and Informatics, Proceedings*, *1*(Wmsci), 2–7.

Benefield, R. (2009). Agile deployment: Lean service management and deployment strategies for the SaaS enterprise. *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, 1–5. https://doi.org/10.1109/HICSS.2009.52

Bibi, S., Katsaros, D., & Bozanis, P. (2012). Business application acquisition: On-premise or SaaS-based solutions? *IEEE Software*, *29*(3), 86–93. https://doi.org/10.1109/MS.2011.119

Boehm, B., & Turner, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, *22*(5), 30–39.

Budgen, D., & Brereton, P. (2006). Performing systematic literature reviews in software engineering. *Int. Conf. Soft. Engin.*, 1051. https://doi.org/10.1145/1134285.1134500

Buford, J., Singh, K., & Krishnaswamy, V. (2015). ALICE: Avaya labs innovations cloud engagement. In *Proceedings of the Principles, Systems and Applications of IP Telecommunications, IPTComm 2015* (pp. 7–14). Association for Computing Machinery, Inc. https://doi.org/10.1145/2843491.2843754

Chamberlain, S., Sharp, H., & Maiden, N. (2006). Towards a Framework for Integrating Agile Development and User-Centred Design (pp. 143–153). https://doi.org/10.1007/11774129_15

Davenport, T. H. (1993). *Reengineering Work through Information Technology*. *Harvard Business School*. Cambridge, MA: Harvard Business School. https://doi.org/10.1177/1354066102008003004

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information. *Information Technolog MIS Quarterly*, *13*(3), 319–340. https://doi.org/10.2307/249008

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2010). The scrum primer version 1.2. *Development*, 1–22. Retrieved from http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf

Femmer, H., Kuhrmann, M., Stimmer, J., & Junge, J. (2014). Experiences from the Design of an Artifact Model for Distributed Agile Project Management. In *2014 IEEE 9th International Conference on Global Software Engineering* (pp. 1–5). IEEE.

https://doi.org/10.1109/ICGSE.2014.9

Grati, R., Boukadi, K., & Ben-Abdallah, H. (2015). SaaS Cloud Provider Management Framework. *Proceedings of the 12th International Conference on E-Business*, 221–228. https://doi.org/10.5220/0005550402210228

Gulledge, T. R. (2006). Erp Gap-Fit Analysis From a Business Process Orientation. *International Journal of Service and Standards*, *2*(4), 339–348. https://doi.org/10.1504/IJSS.2006.010468

Karlesky, M., & Voord, M. Vander. (2015). Agile Project Management (or, Burning Your Gantt Charts), (October 2008).

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. *Information and Software Technology*, *51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Ko, R. K. L. (2009). A Computer Scientist's Introductory Guide to Business Process Management (BPM). *Crossroads, , The ACM Magazine for Students*, *15*(4), 11–18. https://doi.org/10.1145/1558897.1558901

Koski, A., Kuusinen, K., Suonsyrja, S., & Mikkonen, T. (2016). Implementing Continuous Customer Care: First-Hand Experiences from an Industrial Setting. *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, 78–85. https://doi.org/10.1109/SEAA.2016.31

Lauesen, S. (2002). Software Requirements: Styles and Techniques, 591. https://doi.org/10.1016/j.nec.2009.04.003

Lenarduzzi, V., & Taibi, D. (2016). MVP Explained : A Systematic Mapping Study on the Definitions of Minimal Viable Product, (August). https://doi.org/10.1109/SEAA.2016.56

Liu, W., Zhang, B., Liu, Y., Wang, D., & Zhang, Y. (2010). New model of SaaS: SaaS with tenancy agency. *Proceedings - 2nd IEEE International Conference on Advanced Computer Control, ICACC 2010*, *2*, 463–466. https://doi.org/10.1109/ICACC.2010.5486635

Lucassen, G., Dalpiaz, F., Werf, J. M. E. M. Van Der, & Brinkkemper, S. (n.d.). Forging High-Quality User Stories : Towards a Discipline for Agile Requirements.

Luisanna Cocco, Katiuscia Mannaro, Giulio Concas, and M. M. (2011). Simulating Kanban and Scrum vs Waterfall with System Dynamics. *XP 2011: Agile Processes in Software Engineering and Extreme Programming*, pp 117-131.

Mahalakshmi, M., & Sundararajan, M. (2013). Traditional SDLC Vs Scrum Methodology – A Comparative Study, *3*(6), 2–6.

Moens, H., & De Turck, F. (2014). Feature-based application development and management of multi-tenant applications in clouds. *Proceedings of the 18th International Software Product Line Conference on - SPLC '14*, 72–81. https://doi.org/10.1145/2648511.2648519

Moens, H., Dhoedt, B., & De Turck, F. (2016). Management of customizable Software-as-a-Service in cloud and network environments. *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, (Noms), 955–960. https://doi.org/10.1109/NOMS.2016.7502932

Moens, H., Truyen, E., Walraven, S., Joosen, W., Dhoedt, B., & De Turck, F. (2012). Developing and managing customizable Software as a Service using feature model conversion. *2012 IEEE Network Operations and Management Symposium, NOMS 2012*, 1295–1302.

https://doi.org/10.1109/NOMS.2012.6212066

Moens, H., Truyen, E., Walraven, S., Joosen, W., Dhoedt, B., & De Turck, F. (2012). Feature placement algorithms for high-variability applications in cloud environments. *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, 17–24. https://doi.org/10.1109/NOMS.2012.6211878

Mohamed, F., Abu-Matar, M., Mizouni, R., Al-Qutayri, M., & Mahmoud, Z. Al. (2015). SaaS dynamic evolution based on model-driven software product lines. *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, *2015–Febru*(February), 292–299. https://doi.org/10.1109/CloudCom.2014.131

Molina-Jimenez, C., Shrivastava, S., & Wheater, S. (2011). An architecture for negotiation and enforcement of resource usage policies. *Proceedings - 2011 IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2011*. https://doi.org/10.1109/SOCA.2011.6166218

Moore, B., & Mahmoud, Q. H. (2009). A service broker and business model for SaaS applications. *2009 IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2009*, 322–329. https://doi.org/10.1109/AICCSA.2009.5069343

Nasir, M., & Study, C. (2006). A Survey of Software Estimation Techniques and Project Planning Practices, 2–7.

O'Neill, P., & Sohal, A. S. (1999). Business process reengineering a review of recent literature. *Technovation*, *19*(9), 571–581. https://doi.org/10.1016/S0166-4972(99)00059-0

Pajk, D. (2013). Fit Gap Analysis – The Role of Business Process Reference Models, *15*(4), 319–338.

Ruehl, S. T., & Andelfinger, U. (2011). Applying software product lines to create customizable software-as-a-service applications. *Proceedings of the 15th International Software Product Line Conference on - SPLC '11*, 1. https://doi.org/10.1145/2019136.2019154

Schwaber, K., & Beedle, M. (2001). Agile soft-ware development with scrum. NJ, 2003: Up-per Saddle River.

Seethamraju, R. (2015). Adoption of Software as a Service (SaaS) Enterprise Resource Planning (ERP) Systems in Small and Medium Sized Enterprises (SMEs). *Information Systems Frontiers*, *17*(3), 475–492. https://doi.org/10.1007/s10796-014-9506-5

Singhto, W., & Phakdee, N. (2017). Adopting a combination of scrum & Waterfall methodologies in developing tailor-made SaaS products for Thai service & manufacturing SMEs. *20th International Computer Science and Engineering Conference: Smart Ubiquitos Computing and Knowledge, ICSEC 2016*. https://doi.org/10.1109/ICSEC.2016.7859882

Truyen, E., Cardozo, N., Walraven, S., Vallejos, J., Bainomugisha, E., Günther, S., … Joosen, W. (2012). Context-oriented programming for customizable SaaS applications. *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*, 418. https://doi.org/10.1145/2245276.2245358

Wieringa, R. (2014). *Design Science Methodology for Information Systems and Software Engineering*. *Springer Berlin Heidelberg*. https://doi.org/10.1145/1810295.1810446

Zhao, J. F., & Zhou, J. T. (2014). Strategies and methods for cloud migration. *International Journal of Automation and Computing*, *11*(2), 143–152. https://doi.org/10.1007/s11633-014-0776-7

Appendices

Appendix A. Expert Interview

1. What experiences do you have with the implementation of customizable SaaS solutions?
    a. What is the average lead time for an implementation project?
    b. Do you currently use an implementation method?
    c. Have you done implementation projects that were done according to the planning? If they didn't, can you point the problems.
    d. What role plays customizability in your implementations?
2. Is the designed implementation method suitable and complete for implementations of customizable SaaS?
    a. Does the proposed implementation method fit for your implementations?
        i. What steps differ from your current method?
    b. Enables the implementation method you to plan accurately and how?
    c. Is the customer sufficient involved in during the implementation according to the method?
        i. Can the customer involvement vary per implementation?
        ii. Can sufficient customer involvement be guaranteed by the method?
    d. Are the customer requests for customization handled correctly in the implementation method?
    e. Can this method help you in improving your implementation projects?
        i. Which steps or approaches will particularly help in comparison to current used techniques.
    f. Do you have recommendations for adaptions to this method?
        i. With what steps do you agree/disagree and why?

Appendix B. List of Figures

Appendix C. List of Tables