Universiteit Leiden

ICT in Business and the Public Sector

THESIS PROJECT:

Migrating and transforming self-hosted software products into a SaaS solution

Name:	Lennart Timmers
Student-no:	S2133431

Date:	30 April 2021
Version:	Thesis v1.2

1st supervisor:	Dr. Werner Heijstek
2nd supervisor:	Drs. Bas Kruiswijk

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

ABSTRACT

The current IT market is transitioning from utilizing traditional software applications hosted by the customer towards SaaS solutions. This transition is being made to utilize cloud benefits such as scalability, cost reduction and no more maintenance for the customer. This study aims to determine how traditional software vendors can transition their product into a SaaS solution. It specifically investigates which architectural changes need to be made to the software application.

This research makes use of interviews with experts in this field to gather information about the subject. The experts exist out of 2 groups, 5 experts without concrete experience with transitioning to SaaS and 5 experts with concrete experience.

The results showed that in most cases it is smarter to rebuild the application instead of refactoring. The experts indicated that rebuilding will often be less work than refactoring an existing application. The only exception they mentioned was when the application architecture was designed with an eventual move to SaaS in mind. In this case they found refactoring to be a valid option.

ACKNOWLEDGEMENTS

Writing this thesis proved to be quite a challenge for me. There were quite a few up and downs including the switch to full time work next to writing this and corona making it harder to actually find interviewees. I am however content with the final product I managed to produce.

I want to give a special thanks to Werner & Bas who have mentored me through this process. Often taking some time out of their day to help me going through their feedback. This has helped me tremendously.

I also want to thank Xebia and specifically Ellis. Ellis was not only my mentor within Xebia but she also helped me a great deal with handling feedback, giving feedback and sharing her personal experiences. Many of these discussions have helped me to up the quality of this research and thought me about how to handle these processes.

CONTENTS

1- INTRODUCTION

In the past years cloud computing has taken the world by storm (Eric Knorr, 2008). With cloud computing solutions such as Infrastructure as a Service, Platform as a Service (PaaS) and Software as a Service (SaaS) the whole IT architecture landscape grew by having multiple options to host solutions (Cusumano, 2010).

An IaaS solution allows a consumer to pay for and make use of processing, storage, networks and other computing resources provided by the cloud provider (Dillon T, 2010). This means that the customer is responsible for making choices on a technical aspect, they can configure the computing resources for a virtual machine for example and also have to choose the operating system. It allows users to a meet growing or shrinking resource demands from their customers (Mell & Grance, 2011).

A PaaS solution allows a consumer to pay for a development platform supporting the full software lifecycle (Dillon T, 2010). This means that the consumer can deploy and control application on the cloud infrastructure (Mell & Grance, 2011). The added service that is offered by a PaaS is that the consumer does not have to manage the underlying infrastructure such as network, servers, operating systems and storage (Mell & Grance, 2011).

A SaaS solution allows a consumer to pay for a product which is fully maintained on the side of the owner of the software (Eric Knorr, 2008). This means that the consumer does not have to worry about maintaining servers at their own office or installing the software on their work stations (Dillon T, 2010). In addition, SaaS typically employs a subscription model instead of paying a large sum of money once for the full setup of a solution (Stamatia Bibi, 2012). However, these advantages are not limited to the consumer, there are also advantages for the provider. Providers gain advantages such as flexibility, no maintenance on (physical) IT infrastructure and cost reduction since they only pay for services they use (S. Marston, 2011). Fowley et. al. (2017) also mention a number of reasons that independent software vendors (ISV's) have for wanting to move to the cloud. The reasons they report are: cloudification (being able to use cloud advantages), internationalisation, scalability and creating a SaaS product (Fowley, Elango, Magar, & Pahl, 2017). In addition, Marston et. al. also mentions in their article that there could be a form of pressure both from a competitive and trading partner perspective. If either of these switch to a cloud model (S. Marston, 2011).

There are downsides to transitioning to cloud based services his was especially the case in the early days of the cloud software (Shuai Zhang, 2010). The security of cloud solutions used to be insufficient (Shuai Zhang, 2010). Placing sensitive data somewhere in the cloud where it is not clear where it is stored can be a deterrent (Dillon T, 2010). However, the biggest cloud providers Amazon Web Services (AWS), Google Cloud and Microsoft Azure (A. Li, 2010) have increased their

standards since 2010 and have multiple compliance certifications (Microsoft, 2019; Amazon, 2019; Google, 2019). For example, Microsoft Azure has more than 90 compliance certificates, of which 35 specifically aim to satisfy the needs of key industries such as health and government industries (Microsoft, 2019).

Another issue with transitioning to the cloud is the process of reconfiguring existing application to function in a cloud environment without losing any existing data (Cusumano, 2010). This takes time and effort and is an investment for any company transitioning from self-hosted software to the cloud (Cusumano, 2010). Currently most SaaS solutions work with a monthly fee whereas most self-hosted solutions are paid for at once by the customer (Stamatia Bibi, 2012). The effect of this is that the software owner has to switch their revenue model which can be challenging for traditional software vendors (Boillat & Legner, 2013). Switching to another revenue model will have an impact on the business. The business will needs to create a monetization model with income provided by an entirely different revenue model (Fowley, Elango, Magar, & Pahl, 2017).

As discussed above there are a number of issues that companies will face when transitioning from self-hosted solutions towards Software as a Service. However, the current SaaS market is growing (Seethamraju, 2015) as a consequence the pressure to move towards the cloud from competitors and trade partners as mentioned above will also increase (S. Marston, 2011).

In the work of Pahl et. al. (2013) they conclude that there are not any common procedures when executing migrations to the cloud, in addition they also mention a lack of supportive tools. When comparing the works of Mohaghegi & Saether, Fowley et. al. and Pahl et. al. it becomes clear that there is still a lack of knowledge on how to transform legacy application architecture into a SaaS solution. These are general models for moving towards the cloud but none of these mention specifics as to how to transform existing architecture. (Mohagheghi & Sæther, 2011)

2 – PROBLEM STATEMENT

2.1 - Overview

The current market is transitioning from creating and hosting product software which is specifically tailored to their client's needs, towards offering their software as a service through the cloud (Synergy Research Group, 2018). The cloud offers providers advantages such as flexibility, no maintenance on (physical) IT infrastructure and cost reduction (S. Marston, 2011). In addition the pressure providers experience from competitors and trading peers which have transitioned to SaaS makes transitioning towards the cloud enticing.

Providers of product software are looking for a way in which they can transition their current product into a SaaS solution . These steps need to be taken to stay relevant in the market (S. Marston, 2011) (Synergy Research Group, 2018). However, transitioning such a product is a challenge for companies, since there are multiple adjustments which need to be made to their applications (Boillat & Legner, 2013). Just moving the application from an in-house hosting location and configuring it in a cloud environment is not adequate. With this transition the architecture will need to undergo changes to become able to support cloud features (Fowley, Elango, Magar, & Pahl, 2017). This will allow the provider to fully utilize the benefits which are provided by the cloud (Fowley, Elango, Magar, & Pahl, 2017). Benefits such as horizontal scaling and elasticity which allow providers to provide a consistent performance to their customers and to downscale when the load on the application is low, saving them money. These benefits are further explained in chapter 3.

In addition, the business layer will also be impacted by these changes. SaaS services are usually payed for by a pay-per-month plan whereas product software is usually payed for at once (Stamatia Bibi, 2012). This will impact the way that the business will have to handle the business flows as the amount of money available will change (Fowley, Elango, Magar, & Pahl, 2017). However, due to time constraints this change in business flows has been excluded from this research.

2.2 - Research Question

Following the previous chapters, a research question and its sub-questions have been formulated. These questions contain terms which have been mentioned in the chapter above as well. To ensure that it's clear for the reader what these terms mean a definition is provided below.

Product software: Software that is sold as a product by providers to customers. Often this is done by the provider selling the customer a license which allows them to use this software. Once purchased this software is than installed at the customer. This customer is responsible for keeping the software

running, while the provider is responsible for providing updates (security, new features, bug fixes) to the software.

Software as a Service (SaaS): Software that is most often available for customers through a subscription model. The customer pays a monthly fee (usually) and is then able to use the software immediately without having to install this in their own environments. The provider is responsible for keeping the software running and for providing updates (security, new features, bug fixes) to the software.

Main Question

What (types of) changes to the architecture of existing software applications are needed to enable them to be offered as SaaS solutions?

Sub Questions

To answer this question the following sub-questions need to be answered.

1. What differences are there between product software and SaaS applications?
2. What are the key architectural differences between product software and SaaS applications?
3. What changes are required in the life cycle of an application after it has been transitioned to a SaaS solution?

3 – BACKGROUND & SIGNIFICANCE

3.1 - Literature and scientific gap

This chapter discusses the current existing literature and highlights three migration strategies that are found in there.

3.2 - Software architecture

In literature there are multiple definitions as to what software architecture is.
Perry and Wolf (1992) state the following concerning software architecture:
“Software Architecture = { Elements, Form, Rationale }”. They indicate that they believe software architecture is a set of architectural elements (processing, data and connecting) that have a particular form. The rationale is an integral part of the architecture. This rationale motivates the choices for the architectural style that has been chosen (Perry & Wolf, 1992).

Garlan and Shaw (1994) have made the following definition concerning software architecture:

> Software architecture is a level of design that goes beyond the algorithms and data structures of the computation: designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives. (Garlan & Shaw, 1994)

In contrast with the previously mentioned definition by Perry & Wolf (1992), Garlan and Shaw's (1994) definition is broader. In addition to the design aspect which is the main focus of Perry & Wolf (1992), they also mention organization/global control structure and communication protocols, meaning that they chose to include the business aspect in the definition as well.
Philippe Kruchten (2006) states:

"software architecture involves

- the structure and organization by which modern system components and subsystems interact to form systems, and
- the properties of systems that can best be designed and analyzed at the system level."

(Philippe Kruchten, 2006)

In other works (Kruchten, 1995) Kruchten also references the model created by Garlan and Shaw (1994). He agrees with Garlan and Shaw (1994) as he also uses the key words Components and Form in his definition.
One thing that all these theories agree on is that software architecture is on a higher level than the design of the software itself. Software architecture means designing the structure of interacting software components/elements which then end up forming a system.

3.3 - Cloud computing definition

Cloud computing is defined by the NIST and their definition is a market standard in this specific field, it has been cited by more than 17000 other articles on cloud computing. Their definition is as follows:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." (Mell & Grance, 2011)

The NIST definition consists of three key components. These components are *Essential Characteristics*, *Service Models* and *Deployment Models*.

The *Essential Characteristics* they state are On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, Measured service. Below these terms are explained.

- **On-demand self-service**: A user can manage their computing resources without interaction with each service provider.
- **Broad network access:** Available through devices such as phones, laptops and workstations through the internet
- **Resource pooling**: A provider pools their resources serving multiple customers using a multi-tenant model. The customer has some level of control over the location of the resources but only on a high level (I.E. US-West, US-East).
- **Rapid elasticity**: Capabilities can be scaled up or down depending on the demand. This action can be automated.
- **Measured service**: The costs for the customer are determined by monitoring the use of the resources which is visible for both customers and providers. This is usually done through pay-per-use or charge-per-use basis.

The service models consist of: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). For this research the focus is on SaaS and this model will be addressed in chapter 3.5. In the introduction a general explanation of PaaS and IaaS has already been given.

Finally, the deployment models consist of Private cloud, Community cloud, Public cloud and Hybrid cloud.

- **Private cloud:** A private infrastructure exclusively used by a single organization.
- **Community cloud:** A cloud infrastructure for exclusive use by a specific community with shared concerns
- **Public cloud:** A cloud infrastructure available for use by the general public
- **Hybrid cloud:** A combination of two or more different cloud infrastructures that remain unique but are bound by standardized technology.

(Mell & Grance, 2011)

3.4 - Cloud ready/native

Cloud readiness is a term which indicates whether an application is ready to be moved towards the cloud (Loebbecke, Ullrich, & Thomas, 2011). When a legacy application, in this case an application created before the cloud was invented, has to be moved to the cloud it is usually not as simple as just installing it in the cloud. These legacy applications will need to go through multiple steps before they can run and utilize the cloud correctly (P. Mohagheghi, 2011) (Fowley, Elango, Magar, & Pahl, 2017). Once these steps are taken an application can be called cloud ready.

The term cloud native is defined by Kratzke & Quint (2017). They state the following about cloud native applications:

> A cloud-native application is a distributed, elastic and horizontal scalable system composed of (micro)services which isolates state in a minimum of stateful components. The application and each self-contained deployment unit of that application is designed according to cloud-focused de- sign patterns and operated on a self-service elastic platform. (Kratzke & Quint, 2017)

So according to Kratzke & Quint a Cloud-native application needs to be:

- **Distributed**: The application exists out of multiple components which can communicate with each other and function as a single coherent system for the end user.
- **Elastic and horizontally scalable**: The system is able to increase and decrease the computing power automatically based on the demand user place on the application (no degrading performance)
- **Stateful components**: *"used for multiple instances of a scaled-out application component [to] synchronize their internal state to provide a unified behavior"* (Kratzke & Quint, 2017)
- **Elastic platform**: *"is understood as a middleware for the execution of custom applications, their communication, and data storage is offered via a self-service interface over a network"* (Kratzke & Quint, 2017)

The difference between these two terms is that with cloud readiness you are ready to move to the cloud but you do not have to make use of the all the cloud attributes yet. With cloud native applications you are immediately able to make use of these cloud attributes.

3.5 - Software as a Service

SaaS is an aspect of cloud computing as defined by the NIST standards (Mell & Grance, 2011). The NIST definition of cloud computing is a market standard in this specific field, it has been cited by more than 17000 other articles on cloud computing.

The following is a citation of the definition of SaaS according to NIST:

Software as a Service (SaaS). *The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.*

This definition will be used throughout this research when referencing to these terms.

3.6 - Migration approaches

There are a number of studies that address the issue of moving an application towards the cloud (Pahl, Xiong, & Walshe, 2013; Mohagheghi & Saether, Software Engineering Challenges for Migration to the Service Cloud Paradigm On-going Work in the REMICS Project; Boillat & Legner, 2013; Fowley, Elango, Magar, & Pahl, 2017).

The work of Mohagheghi & Saether (2011) discusses the REMICS framework. This framework focuses on the migration of legacy applications to cloud ready applications (Mohagheghi & Sæther, 2011).

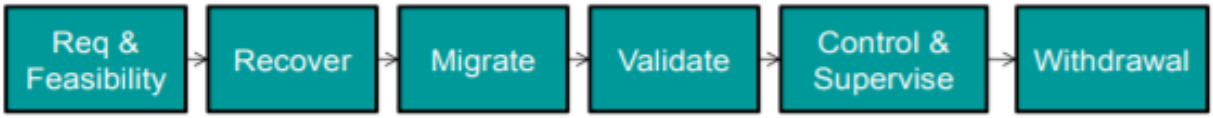

Figure 1 REMICS framework

They have created a general model for migrating such applications consisting of six steps shown above in figure 1. In their work they also discuss how the REMICS approach to a migration looks like. This is shown below in figure 2, which is the same model as figure 1 except it includes the inputs and outputs in between steps. For example, to start the recover phase there has to be a legacy artifact available.

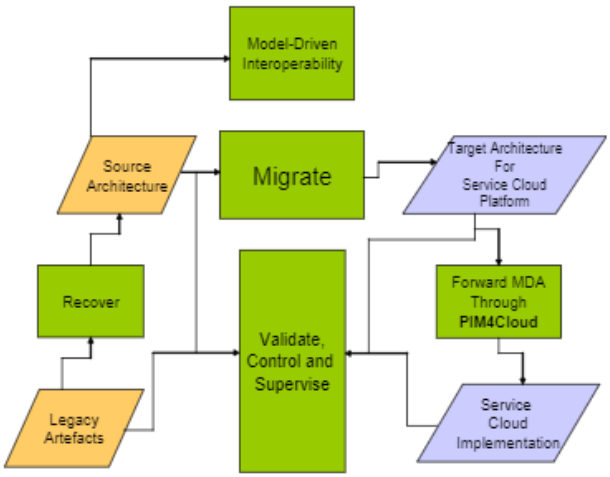

Figure 2 REMICS approach to migration for PIM4Cloud

Mohagheghi & Saether (2011) piloted their work together with one of their partners in the REMICS project as an in-house migration trajectory which mainly focussed on legacy applications.

In the first phase requirement and feasibility, the steps that are taken are validating if it is possible to move the application towards a SaaS solution and if so then determining the requirements that need to be met to do this.

In the second phase, Recover, the REMICS project specifically talks about recovering information such as source code, documentation, execution logs and people's knowledge. Once this information is gathered it will get used to create requirements, architecture, business processes and rules, implementation and deployment models (P. Mohagheghi, 2011).

After these steps it is time to start working on actually migrating the application in the migration step. In the REMICS project they start from the legacy models and refactor them to build a new Service Orientated Architecture (SOA). (P. Mohagheghi, 2011) . In the REMICS project they have a very specific way of migrating by using specific standards such as SoaML a modeling language for the transitioned models, PIM4Cloud and CloudML which they use to abstract the cloud deployment.

After the migration is completed the application has to be validated according to the REMICS framework. The REMICS framework aims to validate that the application corresponds with the old application and has the same or better QoS, business goals and coverage.

Next is the control and supervise phase. REMICS indicates that their goal during this phase is to manage applications by observing them and performing corrective actions. In this phase the provider has successfully completed the migration and is now responsible for running the SaaS solution they have created.

The final phase is called the withdrawal phase. In this phase the REMICS project, which participates as a third party in the migration, stops working on the application.

The study of Fowley et. al. (2017) focuses on independent software vendors (ISV) transitioning software systems to the cloud. They indicate that they use a structured migration process with two core components, A pattern-based approach to determine and analyse migration plans and an early-stage experimentation as a means to address quality and cost considerations.

They state that there are four factors which need to be considered before migration. Setting/Application (description sector and classification application), Expectation/Driver (the vision of the migration benefits the potential users have), Ignorance (Factors that have been overlooked), Concerns (Specific problems/constraints that need to be addressed) (Fowley, Elango, Magar, & Pahl, 2017)

Next they state the following stages in their migration approach (see figure 3): Technology review, Business analysis, Migration & Architecture and Test & Evaluation. However in their research they do not explicitly state how they planned to handle the migration and point to their "*Experimenting*

approach" as their method. They highlight that especially during the Migration & Architecture and Test & Evaluate phase they applied experimentation as migration approach.

Fowley et. al. (2017) indicates that the reason for their experimenting approach is because of a lack of understanding on the side of the ISV. These ISV's do not see the full scale of the impact the change to SaaS will have (difference in provisioning, vendor lock-in), and through this experimenting approach they can identify their architecture options and the costs they will make (Fowley, Elango, Magar, & Pahl, 2017).

Fowley et. al. report on four different cases. These cases were all in house projects which were supported by Fowley et. al. when the companies had to make tough decisions during the migration trajectory.

Technology Review	available technologies and solutions for cloud-based transaction, card and customer storage and processing	available technologies and solutions for cloud-based insurance storage and processing	available technologies and solutions for cloud-based ERP solutions	network concerns for high-speed up/download, services for in-cloud document processing	*Components such as storage, high-performant networks, ERP systems or transaction processing indicate re-engineering focus*
Business analysis	investigate security and monitoring/auditing options for cloud-based banking processing	investigate security and monitoring/ auditing options for cloud-based insurance processing (focus data integrity and location as products offered cross-boarder)	investigate legal (rather than linguistic) localisation requirements regarding the deployment (the ERP system is provided to customers across Europe, but also China)	business analysis to investigate security/data privacy regulations	*Indicates the benefits of cloud-native architectures*
Migration & Architecture	focus on feasibility and efficacy of process-aware migration of banking admin and operations management systems into scalable cloud architecture	focus on implementing business process-aware migration of insurance admin & operations management (policy, accounts, CRM, telephony) into distributed cloud architecture	focus on feasibility and efficacy of process-aware migration of ERP system features (15 core modules) into scalable cloud architecture	development of a 2-staged incremental migration plan (IaaS and PaaS) to migrate a document scanning, storage and processing to scalable cloud architectures.	*Defines the scope of a re-engineering process towards a cloud-native architecture*
Test & Evaluation	evaluate scalability of cloud-based integrated banking service configurations	evaluate scalability of distributed cloud-based integrated insurance service configurations	evaluate scalability of cloud-based integrated ERP service configuration for different markets	Testing of cloud-specific properties: Scalability, Performance, Integration, Security	*Explains why a cloud-native architecture is useful for technical and cost reasons*

Figure 3 - Migration tasks Fowley et. al.

The research of Pahl et. al. identifies common steps to take during migration processes depending on the context of the cloud solution (IaaS, PaaS, SaaS, see chapter 1). According to them they have *"established core elements of a migration process toolkit like standard activities and steps, based on facets of (a here implicit) cloud migration ontology defining major concerns"* (Pahl, Xiong, & Walshe, 2013).

Pahl et. al. (2013) have created a number of images showcasing their migration processes. Below is the process they identified for migration towards a SaaS solution (figure 4).

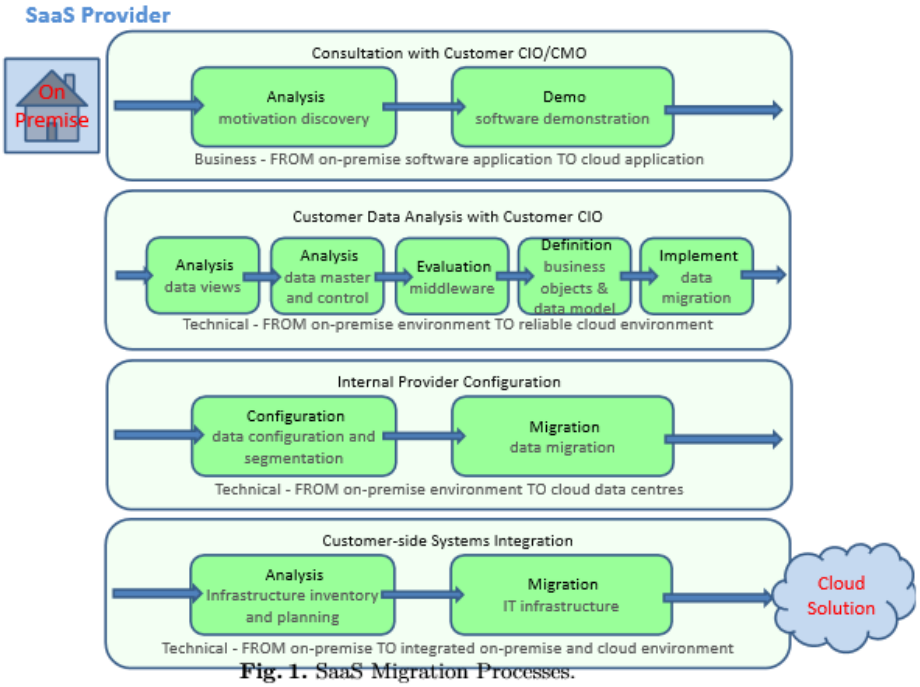

Fig. 1. SaaS Migration Processes.

Figure 4 SaaS migration process as by Pahl et. al.

However, this model focuses on abandoning the current solution and moving towards a readily available solution in the cloud. Therefore, it is not a viable model when a company wants to transform their existing product into a SaaS solution.

Their approach for a SaaS migration is made up of 4 main steps.

- **Consultation with customer:** This step is to discuss the implications that a switch from on premise software to a Cloud solution will bring with it. The goal is to create a motivation on why the customer wants to move to the cloud and to give a demonstration of the software to address all concerns.
- **Customer data analysis with customer:** In this phase the data that needs to be moved is discussed. A selection is made of the current data to ensure only relevant data is being moved and there is no garbage being moved to the new environment. It is determined who will master the data and in addition the application data model will be defined.
- **Internal provider configuration:** This is the act of actually moving the data from the on-premise environment towards the cloud data centre.

- **Customer-side systems integration:** Once the data is available in the cloud it is possible to start moving the applications, which happens here. In this step Pahl et. al. suggest that rarely all on-premise infrastructure will be moved to the cloud and therefore they suggest using a hybrid cloud solution (a mix between cloud an on-premise).

(Pahl, Xiong, & Walshe, 2013)

In their conclusion Pahl et. al. (2013) mentions a plan arising to make a migration pattern catalogue. The reason they give for this idea is to make a more specific process than they currently have stated. Pahl et. al. (2013) have made these models by studying cases of participants in the IC4 migration studies (irish center for cloud computing & commerce).

When comparing all the works mentioned above, it becomes clear that there is still a lack of knowledge on how to transform legacy application architecture into a SaaS solution. It could even mean that there is no one way to approach this as the approach to take might differ per application. There are general models for moving towards the cloud such as the REMICS model of Mohagheghi & Saether (2011), the model of Fowley et. al. (2017) and the model of Pahl et. al. (2013).
These models have some aspects in common for example the first step of a migration project usually exist of some form of analysis, in which there is a check if the application is viable to move to the cloud and which requirements the application has to be able to move to the cloud.
After this step the models all continue with the migration step, in which the move towards the cloud is executed. After migrating the models evaluate the migrated product and test if the solution functions properly in the cloud. This is the final step model of Fowley et. al. (2017) and Pahl et. al. (2013). In the REMICS model they have a dedicated step called withdrawal in which they stop evaluating the application.

To conclude, the above-mentioned models are quite similar in their strategy for moving an application towards the cloud. Their steps are almost identical, and the order is as well. One noticeable difference is that the model of Pahl et. al. (2013) focusses specifically on the different forms of cloud computing (IaaS, PaaS and SaaS) where the REMICS model focusses on legacy applications and the work of Fowley et. al. looks specifically at independent software vendors (ISV's).

Based on the explanation of the current models for transitioning to the cloud state that there is still a lack of knowledge in cloud migration models. The steps of the current models are applicable when moving to the cloud however, these models fail to specify the technical aspects which need to be taken into account. Therefore, there is a need to research how to deal with these missing technical aspects. To expand upon the current models and add this technical aspect which is currently missing.

4 – RESEARCH DESIGN AND METHODS

This research aims to provide knowledge on how to transition a software product that is installed on-premise into a SaaS solution hosted in the cloud. The findings are aimed to be significant for the academic field and to support Xebia and other businesses with transforming software products into a SaaS cloud-based solution.

This chapter will discuss the applied methods to answer the research question and discuss the justification for the design choices that were made. First, the research approach is discussed. In this chapter the selected research method will be discussed. Afterwards the methods for sampling, data collection and data analysis will be expanded upon.

4.1 - Research approach

This research aims to establish a model which can be used for transitioning product software product into a SaaS cloud-based solution. The current research on transitioning to the cloud is already quite extensive on a general level however, this research does only briefly touch on technical changes that will need to be made for the transformation into a SaaS solution. Technical aspects such as how to configure software for multi tenancy and scalability aren't addressed (Pahl, Xiong, & Walshe, 2013; Fowley, Elango, Magar, & Pahl, 2017; Mohagheghi & Saether, Software Engineering Challenges for Migration to the Service Cloud Paradigm On-going Work in the REMICS Project). Therefore, qualitative research will be used. One of the applications of qualitative research is that it can be used to explore substantive areas about which little is known (Strauss & Corbin, 1998), in this case the technical aspect of transitioning to the cloud. Another reason for the selection of qualitative research is that it allows to obtain intricate details about phenomena (Strauss & Corbin, 1998). This research aims to create a model which can be used by companies to execute their transition towards SaaS, obtaining intricate details about how to do this will therefore be a useful addition to this research.

As mentioned before there is only limited research available on transitioning from product software towards a SaaS solution. Therefore, during this research an explorative approach will be used to create a theoretical model. This theoretical model will be created based upon data found by making use of exploratory research methods. According to Brown Exploratory research:

"tends to tackle new problems on which little or no previous research has been done" (Brown, 2006)

This is a good fit with this research since this is exactly what the research aims to do, tackling the problem of how to transition towards a SaaS solution from a technical perspective.

According to Stebbins, exploratory research should make use of methods such as grounded theory (Stebbins, 2001). Grounded theory aims to help describe relationships between concepts and categories (Berthelsen & Frederiksen, 2018). These concepts and categories are discovered through continuous comparison and simultaneous data collection, analysis and coding. In the end this should all integrate with each other to form an emergent theory (Berthelsen & Frederiksen, 2018). The grounded theory method was chosen because it lends itself for uncovering processes (Janet Witucki Brown, 2011). This research aims to discover which processes you have to go through to transition to a SaaS solution which is a good fit with grounded theory, hence it was chosen as method.

Making use of the grounded theory method will help in establishing a model based on multiple individual cases. By combining the specific knowledge gained in the individual cases it is possible to create a general model which will be of use for the specified parties (Xebia and other business) when migrating towards a SaaS solution. Therefore, to establish this model grounded theory will be used

4.2 - Study Sample

To support this study there is a need of experts on the topic of migrating applications towards the cloud (in short cloud experts). Since the research aims to create a new theory, it is necessary that the interviewees are able to provide insights in their thought process on how they would migrate from product software towards a SaaS solution. Therefore, the interviewees must be experts in their field, if they are not this could lead to missed insights either through lack of knowledge or sub-optimal explanations. The information gained from these experts will be used to create the grounded theory. These cloud experts can range in job roles from architects to DevOps engineers and other IT related operations.

The network of Xebia will be used to select a number of candidates to interview. These candidates will consist of a mixture of Xebia employees and contacts from Xebia. The mix of candidates is a precaution against bias to prevent preconceived notions from influencing the research which might be shared between Xebia’s employees.

To gather the sample there will be made use of purposive sampling. This involves pursuing the kind of person in whom the researcher has an interest, somebody that matches the expert description (Thomas, 2017).

This is a non-probalistic way of sampling which means that not all members of the population have a chance of participating in the study (Dudovskiy, 2018). This sampling method has been chosen since there is a limited number of experts available. This is countered by also interviewing externals

however, it is not possible to state with full confidence that the sample is representative of the population.

When working with non-probalistic samples the sample size for semi-structured interviews should be between 5 and 25 (Saunders, Lewis, & Thornhill, 2013). The number of interviews will depend on the results gathered after each interview. If the answers of the participants stay similar and there are no significant changes in answers the interviewing phase will be stopped.

4.3 - Data collection method

To collect data about the subject this research will be using interviews. With interviews a researcher is able to explore views, experiences and beliefs/motivations of individuals on specific matters (Gill, Stewart, Treasure, & Chadwick, 2008). Interviews help when handling a subject where detailed insights from the interviewees is required (Gill, Stewart, Treasure, & Chadwick, 2008). This research aims to create a model which can be of use when transitioning product software into a SaaS solution. To be able to create this model the researcher needs to get a detailed insight in how to do such a transition from the experts. For this reason the choice to conduct interviews was made.

There are three different types of interviews structured, semi-structured and unstructured. For this research the semi-structured interviewing method has been selected. This method allows for a list of topics to be covered during the interview, but also allows topics to be expanded upon when deemed necessary (Thomas, 2017). According to Adams using semi-structured interviews is especially useful when examining uncharted territory and unknown potential issues (Adams, 2015). Since this research aims to explore uncharted territory where the importance of each topic is not yet clear, semi-structured interviews have been selected to collect the data.

For this research the interviews will be held individually on a face-to-face level. If it is not possible to have a face-to-face interview the fallback option will be a telephone interview. The reason as to why face-to-face interviews have been chosen is to be able to watch and listen for nuances of the respondents behavior, which can possibly give useful information about how a respondent feels about a certain topic (Thomas, 2017).

To conduct these interviews the researcher has created a research protocol. This protocol has a few sections as to which topics will be discussed. First, the background of the participant will be discussed. This section consists of a set of questions to determine the background, and the interviewer will only probe a bit when asking about the prior experience if this is deemed necessary.

Next the interviewer will go into the first and second section with substantive questions. In the first section the definition of SaaS and product software is discussed. This section aims to validate the definition found in the literature research and to get a clear understanding on the opinion of the interviewee about this subject. When discussing this the interviewer can use probing if necessary to ensure that the concepts found in the definition of the literature are discussed. This section in combination with the literature aims to answer the first sub-question.

In the next section transitioning from product software towards a SaaS solution is discussed. This section aims to answer the second and third sub-question of this research. In this section there are three questions that aim to discover how to transition from product software towards a SaaS solution. The first question is about when software is ready to be transitioned, the second how to transition this software and the third what has to happen after the transition. These three questions are open ended and therefore the interviewer will probe when necessary, to ensure that the experts share all their knowledge about how to handle these topics.

Finally, the interviewer will ask the participant if they have anything which they would like to add, allowing the expert an opportunity to expand upon or talk about a topic they deem important. Additionally, the interviewer will ask for personal feedback which can be used to improve their interviewing capabilities.

4.4 - Data Collection procedure

Multiple interviews will be conducted, one of these interviews will be a pilot interview. This pilot interview aims to test the interview protocol. Based on this pilot interview it is possible that the interview protocol might be slightly adjusted.

The interviews will be conducted at the location requested by the interviewee, to accommodate them as much as possible. If it is not possible to conduct the interview face-to-face then the interview will be held through a telephone call. Depending on the preference of the interviewee the interview will be held in either Dutch or in English.

All of the interviews will be opened with an introduction. In this introduction the protocol for the interview is explained to the respondent. The respondent is asked for approval for recording the interview and informed that only the researcher and his supervisors will be able to access these tapes. It is also promised that once the interview is transcribed the tape will be deleted.

4.5 - Data Analysis Strategies

All the interviews will be transcribed and analyzed by applying the open-coding principal. A method which supports creation of a grounded theory (Khandkar, 2009; Thomas, 2017). With this method the researcher goes through the transcribed interviews which were collected during the data collection phase. These interviews are then “line-by-line coded” which means that the researcher reads the whole interview and writes down codes at words or sentences in the text. Usually these codes are concepts but it is also possible to use the terminology that the participant uses, which is called “in vivo coding” (Khandkar, 2009).

After the open coding has been completed the next step is to start working on axial coding. In this phase the researcher starts to make links to which codes belong to which other code. In this phase the researcher makes labels for the codes (Thomas, 2017).

Finally, the last phase is the selective coding phase. In this phase the main themes are defined based on core categories and their relationships. The result of this phase will then end up being the basis on which the researcher builds the grounded theory.

5 - RESULTS

The analysis of the data has been conducted as described in the data analysis strategy chapter. First the researcher started analyzing each interview separate from each other. This was done by transcribing the interview and then coding the sections of each interview. After this was done for each separate interview, the researcher wrote down each code in an excel overview.

The result of this procedure was an extensive list with 185 codes. This list contained quite a few similar codes which were adjusted if needed, to prevent duplicate codes with slightly different naming. However, just removing duplicates did not trim the list down to a workable amount of codes. Therefore, it was decided that these codes would be refined before continuing with the axial coding phase. After multiple refinements the total of codes was brought down to 98 codes (see figure below).

API as a SaaS	API as a PaaS	Pay-as-you-go	product suitable for SaaS
Awareness of other clients	architecture overhaul	User management	Innovation
deployment model	Availability	Rewriting source code	one SaaS version
Implementing SaaS	bottleneck	reason for moving to SaaS	exit strategy
business model	customer satisfaction	Scalability	Exposed as API
multi tenancy	customizability	transforming product software into SaaS	application classification
no success stories	data management	lack of information	continious innovation
Negotiations with SaaS providers	data portability	Cultural transformation	technical transformation
Application architecture	DevOps mindset	under/overutilization	Awareness of patches
decision between SaaS and on premise	difference between product and service	software as a product	licensing
Flexibility	different levels of transitioning	responsibility	SaaS security
differences between SaaS and on-prem	Difficult to realize	SaaS definition	lack of technical knowledge
Cloud computing	organizational change	Product software	shaping the product/SaaS
automation	product ownership	Utilized features	Data safety
High workload	product versioning	Subscription management	SaaS Viability
Ease of use	Running SaaS	questioning feasibility	auditability
measurability	SaaS general solution	traceability	product owner
Business process	SaaS provider managed	similarities between product and SaaS	rebuild product as a SaaS
Exposed through internet	SLA	speeding up business process	unburdening customer
Security	User friendliness	Cost model	experience with SaaS
backwards compatibility	observability	SaaS as part of Cloud computing	maintainable/Maintainability
monitoring	marketing	Single tenancy weakest SaaS"].	Challenges with transitioning to SaaS
versioning	development method	Replacing old applications with SaaS solutions	operational excellence
non-technical	product maturity	SaaS priority	no knowledge about the software
SaaS trail	SaaS mindset		

Figure 5 - codes of interviews

Once these 98 codes were reached the decision was made to stop analyzing the codes digitally and to go through each coded interview and note down the most often named codes in each interview using sticky notes. This was done using the list with codes as a base and cross examining these with each separate interview. In the end, this analysis led to an overview with 35 unique codes. Below is a picture of this sticky note analysis.

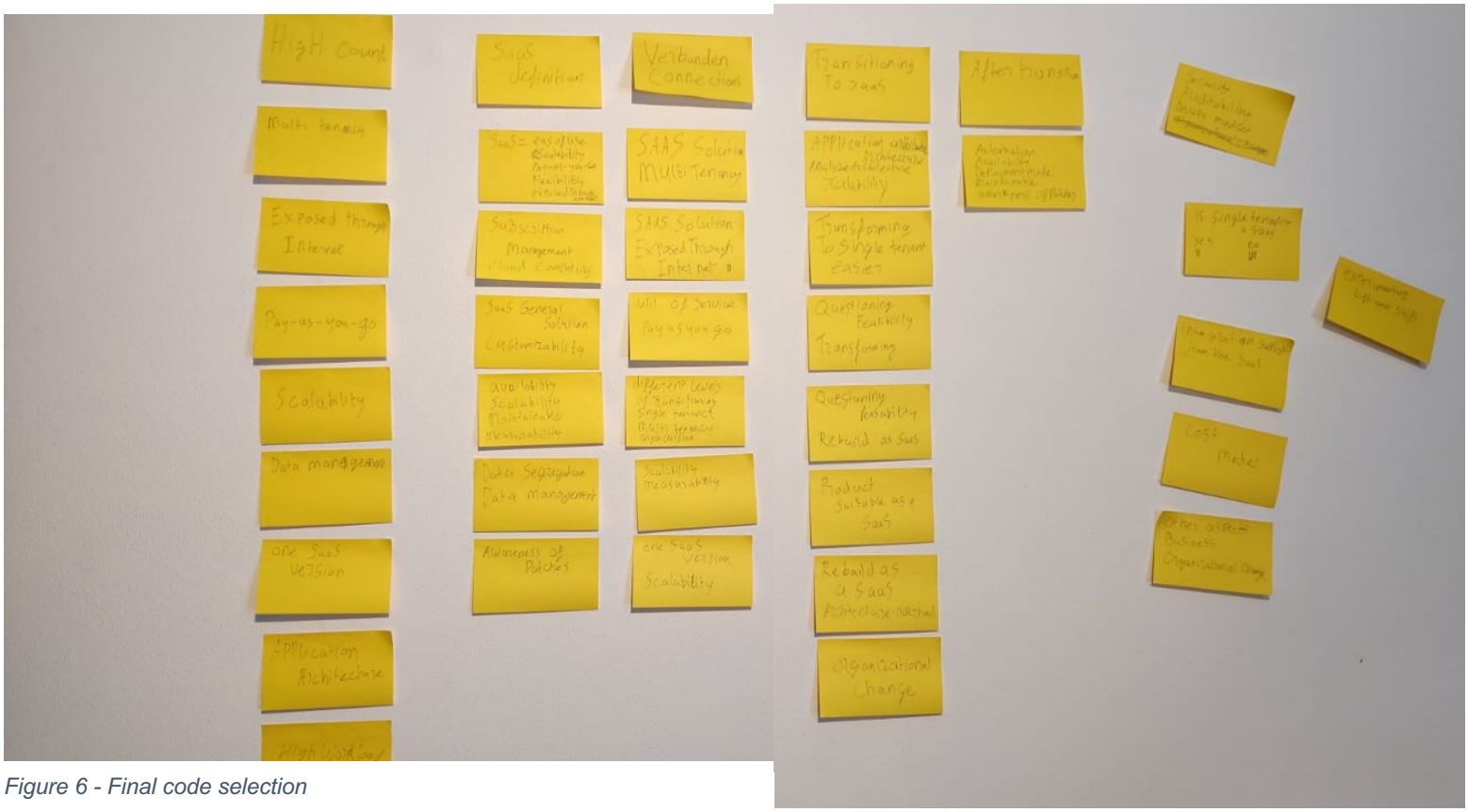

Figure 6 - Final code selection

Based on these 35 unique codes, the researcher then continued analyzing the interviews by continuing with the next step, axial coding. The researcher linked the concepts for each interview separately and wrote these links down on sticky notes as well. Then the transition to selective coding was made. By comparing all the interviews and their contents the researcher created an overview of all the core categories and relationships between them. By writing and sorting these on sticky notes. This led to an overview with the 35 codes and 15 connections (from connections sticky to after transition sticky). Based on these findings and the interview protocol the following points will be presented:

- The meaning and properties of a SaaS terminology
- The terminology of product software
- The differences between SaaS solutions and product software
- When product software is ready for a transition towards SaaS
- Which steps to take during a transition towards SaaS
- What to do after you have transitioned into a SaaS solution

5.1 - SaaS definition

The experts were asked a set of questions to determine what in their opinion represented a SaaS solution. One of the questions asked the participant directly about their definition of a SaaS, while

other questions added more context to the definition by asking about which properties and (non) functional requirements a SaaS solution should have according to them.

The experts have similar opinions on the definition of SaaS, but there were some controversies found with these questions. Each expert mentioned the attributes of multi-tenancy, availability, data management and ease of use/unburdening. Other things that were mentioned by most of the experts were aspects such as scalability, pay-as-you-go models, exposed through the internet, one single SaaS version (Appendix A - table 1). When comparing these results with the NIST essential attributes which belong to each service model (SaaS, PaaS and IaaS) we see the following (NIST definition in italic):

- ***On-demand self-service****: A user can manage their computing resources without interaction with each service provider.* This matches closest with the ease of use but wasn't specifically mentioned by the experts, they do however mention that a customer must be able to configure/customize the application as shown in the quote below:

"that you can configure/customize it to your own liking" – Interviewee 5

- ***Broad network access:*** *Available through devices such as phones, laptops and workstations through the internet.* This is a match with the code exposed through the internet.
- ***Resource pooling****: A provider pools their resources serving multiple customers using a multi-tenant model. The customer has some level of control over the location of the resources but only on a high level (I.E. US-West, US-East).* This is a match with the code multi-tenancy.
- ***Rapid elasticity****: Capabilities can be scaled up or down depending on the demand. This action can be automated.* This is a match with the code scalability.
- ***Measured service****: The costs for the customer are determined by monitoring the use of the resources which is visible for both customers and providers. This is usually done through pay-per-use or charge-per-use basis.* This is a match with the code pay-as-you-go.

There was a disagreement between the experts as to whether you could call a single tenant solution a SaaS solution. Part of the experts (4) were of the opinion that you could do this, while another part (3) was convinced that only a multi-tenant solution could be called a SaaS solution (Appendix A – table 3).
Other aspects that were also were data segregation, customizability, measurability, cloud computing and patch awareness (Appendix A - table 2). Regarding the customizability the experts indicated that this was important to ensure customers had an extensive range of configurable options and to have the option to work with feature flagging to test out new features on a select group of customers.

"So I want to have a a lot of configurable options. And in the end I think that customization and configuration is a bit of perceptions. Because if you can configure a lot then you have the perception that you are able to customize something" – Interviewee 6

“We can enable features on organization level. So within different organizations/tenants we can enable different things. We use this to enable new features in production.” – Interviewee 9

5.2 - Product software definition

The experts did not have too much to say about this and their opinions were similar to each other. The general consensus was that product software is software that is installed on the server of the customer itself. This software is often customizable and specific for that customer. See the following quote by interviewee 6:

“Product software is customizable you can add another layer on top of it. So you can do a lot with it.” – Interviewee 6

Usually it is sold to the customer through a licensing model. Another aspect of this which was mentioned by 5 experts was that if you sell product software you often have to support different versions (Appendix A - table 4). As described by the following quote from interviewee 8:

“With product software … we the supplier are obligated to support an X number of versions” – Interviewee 8

5.3 - Differences between SaaS and product software

This difference is partly deductible as well by using the two definitions given by the experts before. To ensure that no details were missed the interviewer asked the respondents the following question: “*What are the essential differences between product software and SaaS according to you?”*. A quote of one of the interviewees describes product software as follows:

“if you look at the SaaS solution then that is a piece of product software + the infrastructure + exposing the application … So product software is the top layer of that” – Interviewee 4

This quote properly displays the general thoughts of the experts about the differences between SaaS and product software. The difference that the layers mentioned in the quote make are according to the expert’s aspects such as unburdening of the customer, level of customizability and ownership. The expert’s state that the owner of product software is the customer (once they have bought it) and that with SaaS software the provider stays the owner.

The unburdening of the customer comes from the decrease of work a customer has to do to use the application. With product software a customer is responsible for running the software, maintaining the hardware this software is installed upon and upgrading the software when a new version is

released. These tasks all become part of the work the provider has to do with a SaaS solution. Interviewee 7 says the following about ownership:

"If that customer has the software locally in their own database or whatever then the blast radius is only himself, so it is his own responsibility to do everything. Our SaaS proposition then the responsibility is for us, everything concerning security and compliance" – Interviewee 7

The level of customizability decreases when transitioning from product software to a SaaS. Some of the experts expand further upon this by saying that this decrease comes from the need for the SaaS provider to cater to a multitude of customers and that it is therefore not possible to support different product versions, hence the switch to a single version is made.
"And then you have got what I just said as well that you aim to serve with one product multiple people" – Interviewee 3

Another aspect mentioned to change by the experts is the payment model. The most used payment model for product software is a licensing model. With SaaS the experts indicate that this is usually a pay-as-you-go model. Interviewee 6 states the following:

"Most ISV's have licenses for multiple years and with a SaaS license its monthly fees which create discussions around TCO for an application" – Interviewee 6

5.4 - When can software transition to SaaS

To this question the experts answered that the current software would probably not translate to a SaaS solution and it would be easier to rebuild the software from scratch. Some quotes out of the interviews that show this are shown below.

"… And I can imagine that is hard, I think that most product software needs to be reshaped so much that you are better of building it again" – Interviewee 4

"I do not think that when you are looking at a product that you are going to transition this product into a SaaS product" – Interviewee 3

The general consensus is that rebuilding the solution is easier. There is an exception to this according to some of the experts. If the product software initially was created with the idea to transition it into a SaaS then they say it could be possible to refactor the product software's code into a working solution. Interviewee 1 says the following on this subject:

“If that is the case then the ISV has had a good vision towards the future to have configured this in a software product that was installed on one device. I do not think there are many companies that have done this” – Interviewee 1

The demands the experts then state for moving towards SaaS exist of similar requirements that they listed when talking about their SaaS definition. First and foremost, the application should be scalable according to them. The reason that this needs to be present is to prevent unnecessarily spending money on an expensive machine when 90% of the time this machine is not even using 10% of its power.
Another aspect they mention is multi tenancy and data management. The users will sign into the application at the same place as other users. However, the provider needs to ensure that the user can only see their own data. This has to do with the privacy of the customer and the safety of their data (Appendix A – table 5).

Some other things that are not mentioned as much but are noted as important according to experts with transition experience are the readiness of the market for your switch to a SaaS solution and the regulations for running the application in a cloud provider. These regulations usually require certifications from either the provider, cloud provider or both to be in place. Interviewee 8 states the following about the regulations:

“You need to have ISO certificates and these differ between America and Europe … so you need to have your security process in order.” – Interviewee 8

4 of the experts (Appendix A – table 5) also suggest creating an analysis of the current position the product is in before starting a transition. Interviewee 4 & 5 states the following:

“so first you really have to start thinking about the aspects we just named. So what does it mean to offer this product as a SaaS service. So thinking about the functional and non-functional aspects, so an analysis” - Interviewee 4

“Respondent: That depends on what the state is of the product software. and where do you want to go? How scalable are you already, are you able to work with multi tenancy how is your access management? is there a console around that allows you to maintain the product.

Interviewer: so kind of an analysis that allows you to see where you currently are.

Respondent: yes and where do we want to go?” – Interviewee 5

5.5 - Steps to take during transitioning

The experts have a wide range of steps which they would take during transitioning of which some are out of scope for this research as well. The step that is mentioned the most is the rebuilding of the software. As mentioned above this is because refactoring the software is often more difficult than just starting anew according to the experts. They do indicate however that it is probably possible to reuse business logic code. Rebuilding is the main step to take during transitioning. Below are quotes of the interviewees showing their reasons for rebuilding:

“So if they did not start this already then I believe there is a big chance that you have to do with such an old application structure that transforming it from product to SaaS can be such a hassle that I would say take the documentation and start again” – Interviewee 1

“you might have to rebuild your interface into something that does scale and is decoupled better. There is a good chance that the underlying data model has to be redeveloped. But doing that does bring benefits to develop that specifically for a SaaS because only then you can scale in a SaaS way” – Interviewee 2

“I do not think that when you are looking at a product that you are going to transition this product into a SaaS product. I think that the functionality you were offering you are going to offer in a different product” – Interviewee 3

“I can imagine that is hard, I think that most product software needs to be reshaped so much that you are better of starting building it again” – Interviewee 4

A part of this rebuilding step is an overhaul of the architecture. The architecture of the application will need to change to be able to support the new demands mentioned above, the scalability, multi tenancy and data management will all have influence on the new design decisions.

“And also taking a step back, rearchitecting how are we going to do this. How are your deployment journeys and how do you run your application at scale” - Interviewee 6

“scaling up an organization is incredibly hard. Our products have a certain load and that is what they have been made for and if you exceed that everything tumbles down. So you have to start rearchitecting” – Interviewee 7

Other things that are being mentioned by the experts are the organizational changes that will need

to happen during this transition. The experts mention that the provider will probably want to change their business model from licensing to a pay-as-you-go model. They also indicate that the development method that the provider uses might need to change.

"I did experience what it meant for the business so how you sell software changes from licensing to subscription model" – Interviewee 5

"You also notice that developers start working different. You notice that if you make bad software, then it does not take 12 months before that is noticed. That takes half an hour, which is confronting, maybe test a bit more, do TDD" – interviewee 7

5.6 - Steps to take after the transition

Once the provider has transitioned the experts expect them to be able to maintain the application. To do this the provider needs to take multiple steps, some aspects that are mentioned multiple times are automation, lack of knowledge, continuous integration and delivery, development model and versioning (Appendix A – Table 6).

The versioning can become a single version according to the experts, there is no need for supporting multiple versions anymore since the provider is now responsible for running the application and therefore can make this choice. This will make it easier for the provider since they now have to support only one version instead of multiple. Interviewee 8 indicates this in the quote below:

"With product software … we the supplier are obligated to support an X number of versions … with SaaS this is way simpler you can say this is it, you can reach it here. We maintain this as experts." – Interviewee 8

The lack of knowledge mentioned by the experts refers to the change to a whole new architecture. The people within the providers organization might not have the necessary knowledge to be able to develop in this new architecture and if this is the case they will need to either hire new knowledgeable employees or retrain their current employees. This lack of knowledge can form a hurdle when trying to maintain the application. Interviewee 3 states the following:

"But for a lot of companies that is a total Mindshift and a big challenge. You have to start hiring, educate people in a different way maybe start judging in a different way" – Interviewee 3

The experts state that the development model will change since SaaS solutions allow for quicker development methods instead of a slow release cycle that is often found according to the experts

within product software. With a SaaS solution the provider is not dependent on the customer to upgrade their software to newer versions. This allows the provider to make use of faster development methods including practices such as automation and continuous integration and delivery. This will help the provider to work more efficiently when maintaining the software.

"Our intention is to switch to a continuous delivery model, fail fast model. So we just push the software and when it fails we should react quickly, bugfix, and role this fix out to our customers fixing the issue quickly" – Interviewee 10 on their transitioning

"I think the mindset shift you have to make from waterfall to agile …" – Interviewee 9

6 - DISCUSSION

This chapter discusses the findings, which were presented in the results chapter, in a broader context. It discusses the requirements a SaaS solution has to meet and which of these requirements are up to the opinion of the implementor. Furthermore, it expands upon existing theories as discussed in the background and significance chapter. Lastly the limitations of the research and follow up actions are discussed.

6.1 - Contributions of the research

Requirements of a SaaS solution

The Cloud and SaaS definition that is most often used is the definition that is provided by the NIST (> (over 15.000 citations) (Mell & Grance, 2011). In this research the results showed that this definition is indeed complete and correct, see the comparison in chapter 5.4. However, there was an interesting disagreement between the experts. Some of them stated that according to them a SaaS solution does not have to be multi-tenant to be called a SaaS solution, while others specified that this was a necessary requirement for them. The latter one is also in accordance with the NIST definition.

The explanation that these experts gave for their standpoint came down to the point of view. They admitted that to the supplier this solution might not qualify as a SaaS solution, depending on their demands of a SaaS solution. However, for a customer the solution can be perceived as a SaaS solution, they might even be happy in accordance to the experts because their data is not in the same locations as their competitor's data. Their argument is that because the customer can still gain all the advantages of a SaaS solution, they will not care for the lack of multi tenancy.

As mentioned by one of the experts a provider usually wants multi tenancy to ensure that they can optimize their spending on resources. However, this is not necessary to satisfy their users, so the provider has to make a choice when transitioning to SaaS between a single- and a multi-tenant solution.

Expansion upon existing research

The goal of this research was to discover what needed to be done with software architecture and development flow when transitioning this into a SaaS solution. The existing research about transitioning to SaaS is currently quite general, as in the technical handlings that need to be done are not specified. This research has explored the discovery of what the technical handlings are that need to be done. Since applications differ in many ways this will still be a generally applicable model, however it will include the general technical steps that need to be taken during the transitioning.

In the background chapter there were a number of models discussed for transitioning to SaaS, this chapter will be expanding upon the REMICS framework, since this framework has the concrete model and has the most in common with the findings in this research. The commonality is especially found in the steps that this research also found. Each step from the REMICS framework can be related to a step found in this research which will be done below.

The REMICS framework exists of 6 phases, these phases will be discussed and expanded upon. In some cases the research will also propose to alter this model in certain ways since the REMICS framework was being used specifically for legacy applications.

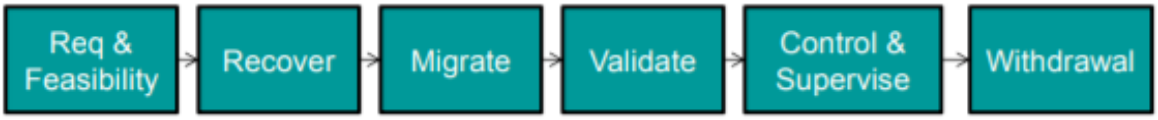

Figure 7 - REMICS framework

During the first phase there is a check on the current application and whether it's possible to move it to the cloud. The only addition this research proposes to this phase is to include the decision making about multi tenancy here as mentioned above. Ensuring this choice is made before the act of actually migrating the application is important because it will heavily influence design decisions for the transitioned application.

Since the REMICS project specifically focused on legacy applications the second phase was specifically named the recovery phase, if the application that has to be moved is still in active use this will be less necessary since there are most likely developers working on it and existing documentation which can be used during this phase. In that case it is easier to combine the first two steps into one step.

Next up is the step of migrating the application. REMICS proposes to use specific techniques and technologies for this however during this research there were no specific mentions of using specific technology, models or tools to migrate. Therefore, this research proposes to leave the selection for these technologies up to the party that is executing the transition. This will remove a limiting factor from this party and allow them to work with familiar models/technologies/tools.

This research does agree with the REMICS framework that the first step in the migration should be the creation of a new architecture for the application. The technical details that should be considered during the designing of the new architecture are scalability, multi-tenancy and data segregation. These are the most important but not the only requirements the architecture should abide by. The general consensus is that the architecture should comply with the requirements of the SaaS definition. This is also the point that the decision which was made in the previous steps between multi-tenancy and single tenancy has a big impact on the design. If the choice for single tenancy was made by the provider, then the number of requirements that need to be met will decrease. For example, there is no need to include segregation into the design since applications will not share multiple users.

The next action to take would be to actually migrate the application. As stated in the results chapter this will most likely mean that the provider has to build a new application based on this new architecture that was created. In some edge cases it might be possible to refactor the existing application into a SaaS solution, however this is usually only the case if in the initial application design an eventual move towards SaaS was accounted for. So technically this will mean that the development team will redevelop/refactor the application using the new architecture standard. Usually the business logic can be recycled from the old application since this will not change, the changes will happen in aspects which will connect to the cloud and aspects which expose the application to the users.

Another thing to take into account during this migration phase is that the provider might also want to have a look into switching up their development model. This was mentioned by multiple experts and one of them explained the reasoning for this was that product software is usually only released about once in every three months. This is mainly because the customer does not want to be bothered by having to frequently install updates. However, once the customer is migrated towards the SaaS solution the responsibility for updates moves to the provider. This means that they can decide to start releasing more and possibly adapt an agile way of working which could be depending on the provider beneficial by providing value quicker than before.

Next up is the validation phase. Testing the application is of course necessary but one thing found during this research was to not just test the application as a provider. In their interviews the experts proposed asking some customers to test the application for you by becoming early adapters. This offers the early adapters the benefits of SaaS and helps the provider to search for issues which managed to stay undetected during their own testing phase.

After the validation phase the control and supervise phase comes next. If the product application was maintained by a team from the customer, then this will be a new responsibility for the provider. This is something that is mentioned by the experts to take into consideration.

In this research it was found that the new responsibilities will include maintaining the platform for the application and ensuring that the application runs properly and providing customer support. Providers need to ensure that their organization is ready to take responsibility for these tasks, this can be made easier according to the experts by implementing best practices such as automation, CI/CD and a single version. These practices all decrease the amount of work a person has to do manually and therefore creates more time to be spent on task that demand manual attention.

The withdrawal phase is the final phase in the REMICS model. This step is not of relevance when a provider is executing the migration for their own product, but it might be if the party migrating it is doing this as an assignment for a customer. However, for the provider the steps will end with the control and supervise phase since they will stay responsible for running the application.

Now that all the steps have been discussed the following changes are proposed to the framework:

- Include a decision between single and multi-tenancy during the requirements & feasibility step
- Merge the requirements & feasibility step with the recovery step
- Free selection of methods/technologies during the migration step
- Decide between rebuilding and refactoring the application during the migration step based on the new architecture design
- Validating the applications functionality with users during the validation step
- Include recommended best practices for running and maintaining the application in the control and supervise phase.
- Clarify that the withdrawal phase is optional and that for the provider the last phase will be the control and supervise phase.

If you incorporate these changes into the current REMICS framework then it will expand and look like the figure shown below.

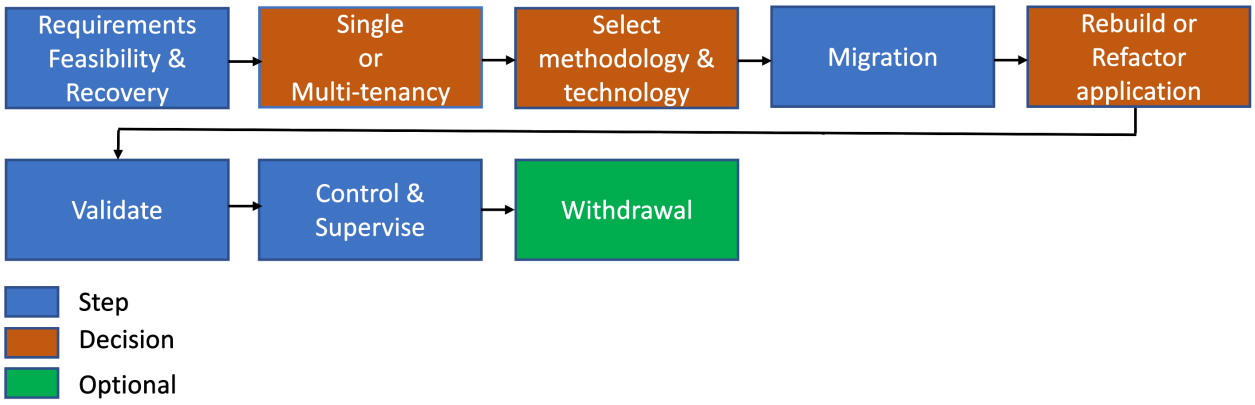

Figure 8 - proposed expansion on REMICS framework

In the figure the requirements and feasibility step have merged with the recovery step. The figure now also includes decision points for single or multi-tenancy, selection of methodology and technology and rebuilding or refactoring. During the validation step the application will not just be validated by the provider but also by a select group of their users. During the control & supervise step it is recommended to start implementing best practices such as automation, CI/CD and a single version to optimize the time engineers can spend developing. Finally, the withdrawal phase has become optional depending on who is using this model. If the model is used by an external party that executes the migration, then they can include this withdrawal step.

6.2 - Limitations of the research

During this research there were a number of limitations. These limitations may have had impact on the research. In the chapter “Research design and methods: study sample” it is mentioned that the interviews will consist of a mixture of employees of the firm Xebia, at which the research was conducted, and IT experts outside of Xebia. The contacts which were interviewed did however still share some commonalities. Firstly, all the respondents were Dutch within a similar expertise field namely IT architecture. However, since the opinion of experts was required it was decided to only interview in that specific expertise field. Even though this is the case, it should be acknowledged that the similarity in background might have had influence on the results. Another commonality is that all the interviewees have some form of connection to Xebia, because the Xebia network was used to find interviewees which met the requirements.

The result of this research, the grounded theory, can therefore not be presented as a sound theory since there is a chance of bias based upon the sample. However, the theory can prove useful for organizations which are aiming to transform their product software into SaaS software.

7 - CONCLUSION

In the current market companies are adapting towards a cloud-first strategy to utilize the benefits the cloud brings with it (Kundra, 2011). This change in adaptation leads to a change in demand in the IT market, since traditional product software does not comply with this strategy. To stay relevant the IT market needs to change their product software into a SaaS solution. However, this is not as simple as lifting and shifting the application from an on-premise hosting solution into a cloud solution, as the advantages of the cloud cannot be utilized if the application gets transitioned with this approach. Companies need to make changes to the architecture of their application before they can move to and successfully utilize the cloud and its advantages.

7.1 - Findings

The main finding of this thesis is that when a company wants to transition their existing software application into a SaaS solution, it is often smarter to start over and create a new application instead of rearchitecting the existing solution. The experts state in interviews that it is possible in some cases to rearchitect but only if in the original application this was taken into consideration. Companies can reuse existing parts of business logic according to them, but rearchitecting the whole application will be more work than starting anew.

Other findings are the divide between single- and multi-tenancy, and what to do after transitioning towards the cloud. The experts could not agree on whether a single tenancy solution could be called a SaaS solution. Therefore, this research concludes that a provider should decide for themselves if a single tenant solution is sufficient for them. After the transition the experts mention that the provider is now the responsible party to maintain the application on the hosting side as well. They suggested to make use of automation, continuous integration and delivery and possibly to change the development model.

7.2 - Future research

In order to validate the new theory presented in this thesis it is recommended to conduct a follow-up study. This study should include more participants with different backgrounds to prevent bias as best as possible.

A point which often came into discussions during interviews was the business perspective when undergoing such a transition. Participants indicated that not only the technical processes within companies would change but also the business processes. This ranged from the way the product would be marketed to the change in the way payments would be made for the product, as SaaS providers often use pay-as-you-go models. However, this change in the business processes was out of scope for this research and has therefore not been explored further. A follow-up study on this subject would provide a wider view on the subject of transitioning product software into a SaaS application. The combination of this research and the aforementioned follow-up studies can prove useful for companies undergoing a transition to SaaS, as it will provide them with a relevant insights and action points for their transition.

BIBLIOGRAPHY

Philippe Kruchten, H. O. (2006). *The Past, Present, and Future of Software Architecture.* IEEE SOFTWARE.

Perry, D., & Wolf, A. (1992). *Foundations for the Study of Software Architecture.*

Pahl, C., Xiong, H., & Walshe, R. (2013). A comparison of on-premise to cloud migration approaches. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8135 LNCS*, pp. 212-226.

Mohagheghi, P., & Saether, T. (sd). *Software Engineering Challenges for Migration to the Service Cloud Paradigm On-going Work in the REMICS Project.*

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing.* National Institute of Standards and Technology, Gaithersburg, MD.

Kruchten, P. (1995). *Architectural Blueprints-The "4+1" View Model of Software Architecture.*

Kratzke, N., & Quint, P. (2017, 4 1). Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study. *Journal of Systems and Software, 126*, 1-16.

Khajeh-Hosseini, A., Greenwood, D., & Sommerville, I. (2010). Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. *2010 IEEE 3rd International Conference on Cloud Computing*, (pp. 450-457).

Garlan, D., & Shaw, M. (1994). *Characteristics of higher-level languages for software architecture.* Pittsburgh: cmu software engineering institute.

Frey, S., & Hasselbring, W. (sd). *An Extensible Architecture for Detecting Violations of a Cloud Environment's Constraints During Legacy Software System Migration.*

Fowley, F., Elango, D., Magar, H., & Pahl, C. (2017). Software system migration to cloud-native architectures for SME-sized software vendors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 10139 LNCS*, pp. 498-509. Springer Verlag.

Contreras, L., López, V., De Dios, Ó., & Fernández-Palacios, J. (2012). Towards cloud-ready transport networks. *10th International Conference on Optical Internet, COIN 2012*, (pp. 91-92).

Boillat, T., & Legner, C. (2013). From on-premise software to cloud services: The impact of cloud computing on enterprise software vendors' business models. *Journal of Theoretical and Applied Electronic Commerce Research, 8*(3), 39-58.

Bibi, S., Katsaros, D., & Bozanis, P. (2010). Application development: Fly to the clouds or stay in-house? *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, (pp. 60-65).

Cusumano, M. (2010, April). *Technology Strategy and Management Cloud Computing and SaaS as New Computing Platforms*. Opgeroepen op 9 19, 2019, van ebusiness.mit.edu:

http://ebusiness.mit.edu/research/papers/2010.04_Cusumano_Technology%20Strategy%20and%20Management_273.pdf

Eric Knorr, E. A. (2008, April). *What Cloud Computing Really Means.* Opgeroepen op 9 19, 2019, van skysolutions: http://skysolutions.co.zw/docs/What_Cloud_Computing_Really_Means.pdf

Dillon T, E. A. (2010). *Cloud Computing: Issues and Challenges*. Opgeroepen op 9 19, 2019, van sistemas-humano-computacionais: http://sistemas-humano-computacionais.wdfiles.com/local--files/capitulo%3Asistemas-de-servico/Dillon2010.pdf

Stamatia Bibi, E. A. (2012). *business application acquisition: On-premise or SaaS-based solutions?* Opgeroepen op 9 19, 2019, van Revistas: https://revistas.um.es/gesinfo/article/download/207621/239171/742211

Shuai Zhang, E. A. (2010). *Cloud Computing Research and Development Trend .* Opgeroepen op 9 19, 2019, van amazonaws: https://s3.amazonaws.com/academia.edu.documents/7299069/05431874.pdf?response-content-disposition=inline%3B%20filename%3DCloud_computing_research_and_development.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20190922%2Fus-ea

Microsoft. (2019). *https://azure.microsoft.com*. Opgeroepen op 9 19, 2019, van Azure compliance: https://azure.microsoft.com/en-gb/overview/trusted-cloud/compliance/

Berthelsen, C. B., & Frederiksen, K. (2018). A Comprehensive Example of How to Conduct a Literature Review Following Glaser's grounded Theory Methodological Approach. *International Journal of Health Sciences*, 91-100.

Charmaz, K. (1996). Grounded Theory. *Rethinking methods in Psychology*, 27-49.

Thomas, G. (2017). *How to do your research project.* Glasgow: Sage.

Dudovskiy, J. (2018). *The Ultimate Guide to Writing a Dissertation in Business Studies: A Step-by-Step Assistance.* Pittsburgh: Research-Methodology.net. Opgehaald van Research Methodology: https://research-methodology.net/sampling-in-primary-data-collection/non-probability-sampling/#_ftnref1

Saunders, M. N., Lewis, P., & Thornhill, A. (2013). *Research methods for Business students.* London: Pearson.

Gill, P., Stewart, K., Treasure, E., & Chadwick, B. (2008). Methods of data collection in qualitative research: interviews and focus groups. *BRITISH DENTAL JOURNAL*, 291-295.

Khandkar, S. H. (2009). *Open Coding.* Calgary: University of Calgary.

Amazon. (2019). *AWS Compliance*. Opgehaald van Amazon: https://aws.amazon.com/compliance/

Google. (2019). *Compliance resource center*. Opgehaald van Google: https://cloud.google.com/security/compliance/

Beserra, P., Camara, A., Ximenes, R., & et. al. (2012). Cloudstep: A step-by-step decision process to support legacy application migration to the cloud. *2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, MESOCA 2012* (pp. 7-16). Piscataway: IEEE.

Loebbecke, C., Ullrich, T., & Thomas, B. (2011). *Assessing Cloud Readiness: Introducing the Magic Matrices Method Used by Continental AG.* Berlin: Springer.

Synergy Research Group. (2018, 8 21). *Quarterly SaaS Spending Reaches $20 billion as Microsoft Extends its Market Leadership*. Opgehaald van srgresearch: https://www.srgresearch.com/articles/quarterly-saas-spending-reaches-20-billion-microsoft-extends-its-market-leadership

P. Mohagheghi, E. A. (2011, June 7). *SINTEF+S20069.pdf.* Opgehaald van sintef.brage.unit.no: https://sintef.brage.unit.no/sintef-xmlui/bitstream/handle/11250/2430591/SINTEF+S20069.pdf?sequence=1

A. Li, E. A. (2010). *CloudCmp: Comparing Public Cloud Providers.* Opgehaald van Microsoft: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/cloudcmp-imc2010.pdf

S. Marston, E. A. (2011). Cloud computing - The business perspective. *Decision Support Systems*, 176-189.

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*, 475-492.

Mohagheghi, P., & Sæther, T. (2011). Software engineering challenges for migration to the Service Cloud Paradigm: Ongoing work in the REMICS project. *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011* (pp. 507-514). IEEE.

Strauss, A. L., & Corbin, J. M. (1998). Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory. In A. L. Strauss, & J. M. Corbin, *Basics of Qualitative Research* (pp. 11-12). Sage Publications, Inc.

Brown, R. (2006). Doing Your Dissertation in Business and Management: The Reality of Research and Writing. In R. Brown, *Doing Your Dissertation in Business and Management: The Reality of Research and Writing* (p. 43). Sage Publications.

Stebbins, R. (2001). *Exploratory Research in the Social Sciences. .* Thousand Oaks: Sage.

Adams, W. C. (2015). Conducting Semi-Structured Interviews. *Handbook of Practical Program Evaluation, Fourth Edition*, 492-505.

Janet Witucki Brown, S.-l. C. (2011). Becoming an Older Volunteer: A Grounded Theory Study. *Nursing Research and Practice*, 2.

Kundra, V. (2011). *Federal Cloud Computing Strategy.* Washington: The White House.