Cloud migration

Master of Science in Technology Thesis
University of Turku
Department of Future Technologies
2021
Jere Lehtinen

UNIVERSITY OF TURKU
Department of Future Technologies

JERE LEHTINEN: Cloud migration

Master of Science in Technology Thesis, 55 p.
January 2021

Migrating on-premises applications to cloud environment has become a popular task for organizations. In this thesis, cloud migration is defined to be an action where one or more parts of an application are migrated to a cloud platform. Multiple motivations are mentioned for such migration like reducing costs, flexibility, and scalability of the application. This thesis also goes through different strategies for cloud migration. After that, a literature-based, generalized migration process for cloud migration was created. This created migration process was then validated against case process.

Phases in the case process were investigated through interviews. Interviews were done in two parts. First, all interviewees were interviewed one at a time. From these interviews, a draft of the case process was done. This draft was then validated and supplemented with a group interview.

After creating both processes, they were compared. It was found that the literature-based process had a lot of similarities with the case process. Also, it was found that the case process had a few tasks that were not mentioned in the literature-based process. These tasks were discussing future of the application, estimating workload and project end date, defining migration scope, and familiarizing customers with application. These can be said to be important tasks, and they should have been in the literature-based process too.

Keywords: cloud computing, cloud environment, cloud migration, cloud migration strategies, cloud migration process

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos

JERE LEHTINEN: Cloud migration

Diplomityö, 55 s.
1 2021

Pilviympäristöjen kehittyessä asiakkaan omalle palvelimelle asennettujen ohjelmistojen migraatio pilviympäristöön on muodostunut suosituksi toimenpiteeksi. Tässä tutkielmassa pilvimigraatio määritellään olevan toimenpide, jossa yksi tai useampi osa ohjelmistosta siirretään käyttämään pilvialustaa. Pilvimigraatio tuo monia etuja, kuten kustannusten väheneminen, sekä ohjelmiston joustavuus ja skaalautuvuus. Tässä tutkielmassa esitellään myös erilaisia strategioita pilvimigraatioon. Tämän jälkeen tutkielmassa esitellään kirjallisuuteen perustuva yleistetty pilvimigraatioprosessi. Tämä yleistetty prosessi vahvistetaan esimerkkitapauksella.
Esimerkkitapauksen vaiheet on selvitetty kyselytutkimuksella, joka koostui kahdesta osasta. Ensimmäisessä osassa jokainen haastateltava haastateltiin erikseen. Ensimmäisen kierroksen jälkeen oli mahdollista tehdä luonnos esimerkkitapauksen vaiheista. Tätä luonnosta vahvistettiin ja täydennettiin kyselytutkimuksen toisella kierroksella, missä kaikki haastateltavat kutsuttiin samanaikaisesti haastateltavaksi.
Lopuksi yleistetty prosessi, sekä esimerkkiprosessi olivat selvillä ja niitä vertailtiin keskenään. Tämän vertailun tuloksena huomattiin, että prosessit muistuttivat paljon toisiaan. Esimerkkiprosessissa oli myös joitain vaiheita, joita ei ollut kirjallisuudessa huomioitu. Nämä vaiheet ovat sovelluksen tulevaisuuden suunnittelu, migraation työmäärän ja päättymispäivämäärän arviointi, migraation laajuuden määrittely, sekä uuden sovelluksen esittely asiakkaalle.

Asiasanat: pilvilaskenta, pilviympäristö, pilvimigraatio, pilvimigraatiostrategiat, pilvimigraatioprosessi

Contents

1 Introduction

In the early stages, the standard way to deploy web-based systems was to deploy them on-premises of the user organization. The process of buying software started by buying one or more server machines. After that, the new application could be installed on those servers. The organization also had to buy perpetual software licenses and personnel to install and maintain servers and software in them. With bought servers, organizations got a fixed amount of computation power. When the organization needed more computation power, they had to buy more server machines.

On-premises environment has some problems that are not present in cloud environment. For example, with cloud environment, it is not needed to buy server machines. Resources that the organization needs from the cloud are paid and bought by their needs. Especially smaller companies and other entities that cannot invest in their own hardware benefit from this arrangement. Also it is easy to monitor and scale resources in cloud environment.

Deploying web-based systems to cloud environment has become a standard way to deploy applications, because of the benefits that the cloud offers. Besides deploying new applications to the cloud, migrating old on-premises applications to cloud has become popular too. After doing such migration, applications will also gain the benefits that cloud environment offers. Migration process must be well-defined and feasible for companies since cloud migrations have become popular.

The object of this thesis is to view different migration strategies and to create a generalized description of migration process from the literature. Generalized description of migration process is important due to increasing interest in cloud migrations. Created description is then validated by comparing it with migration that Aveso did when migrating their on-premises software. Based on this object, three research questions are presented. Research question (RQ) 1: What tasks have to be done in cloud migration process? RQ2: What different types of cloud migrations there are and how they affect the architecture of the application? RQ3: Which cloud migration types benefit from model process?

Cloud computing and on-premises environment are described in chapter two. After explaining these environments, their benefits and challenges are compared with each other. Chapter three goes through what cloud migration is and why it should be done. After that, it explains what strategies there are for migration and what is the migration process like. Fourth chapter introduces case and research method. Fifth chapter explains everything that was done during case migration, its benefits, and its specialties. In sixth chapter, case process is compared with general migration process which was described in chapter three. After that, changes in case architecture are compared with migration strategies that were also introduced in chapter three. Chapter seven contains conclusions of the work.

2 Web-based systems

Web-based systems are systems that are accessed through web browser. These systems can have a lot of functionalities that are controlled with a browser. For example, there are web-based CRM systems, web-based Microsoft Office, and more. With web-based systems, users don't have to install client native applications on to their devices, since these systems are accessible by using only browser.

Web-based systems can be hosted in different ways. In this thesis, we only focus on two different environments. These environments are cloud and on-premises.

2.1 Cloud computing

First virtual machines were created by IBM in the 1970s but the term *"cloud"* was not introduced until the 1990s by telecom companies. In 2006, Amazon introduced its cloud services and Google introduced Google Docs. Cloud computing has become the standard way to implement web-based systems nowadays. Cloud computing has experienced some changes during its lifetime to get to this position where it stands now. [1]

The modern definition for cloud computing is from the National Institute of Standards and Technology (NIST): "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. This cloud model is composed of five essential characteristics, three service models, and four deployment models." [2]

This chapter is going to go through those essential characteristics, service models, and deployment models that are mentioned in the definition of cloud computing. There exist multiple cloud computing service providers, thus Microsoft's Azure will be presented as an example.

2.1.1 Essential cloud characteristics

NIST definition for cloud computing mentions five essential characteristics of cloud. These characteristics are listed here.

1. On-demand self-service. This characteristic means that user or customer has the access to the service without anyone manually granting it.

2. Broad network access. This means that user should be able to access the service without a large amount of bandwidth.

3. Resource pooling, which means that users do not have dedicated resources, since they will not need all their available resources simultaneously. This means that unused resources can be used by another user, and so the provider can serve more users at the same time.

4. Rapid elasticity. Rapid elasticity means that cloud service needs to grow easily and rapidly, whenever user demands or needs it.

5. Measured service, this means that the usage should be able to measure. This can be achieved, for example by measuring usage time.

2.1.2 Cloud service models

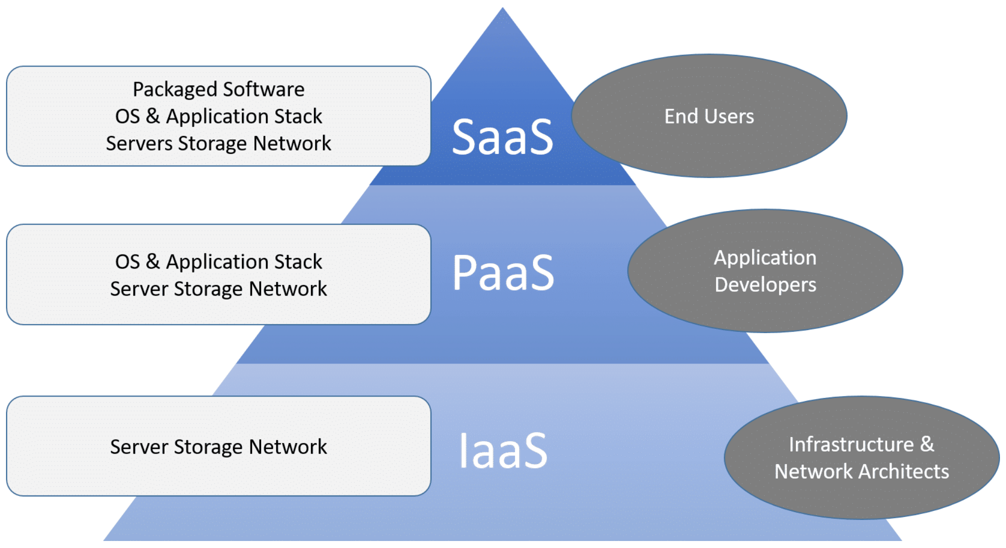


Figure 2.1: Cloud service models [3]

Service models are probably the best known by their abbreviations: SaaS, PaaS, and IaaS. These abbreviations stand for Software as a Service, Platform as a Service, and Infrastructure as a Service, respectively. Figure 2.1 presents these three layers in order of the available control. On SaaS-layer user has almost no control over the application whereas on IaaS-layer user has the most control. SaaS-layer is usually meant for end-users, whereas IaaS is meant for IT administrators. Service models are also presented in [2].

1. Infrastructure as a Service is the lowest layer of cloud services. IaaS provides direct access to the provider's hardware, virtual machine, networking, or storage. When utilizing IaaS, one or more of the mentioned services are used.

2. Platform as a Service is a service that provides the operating system, storage, database, and middleware for development, like Java or .Net Runtime. PaaS provides an easy way to develop applications since you do not have to think about the infrastructure.

3. Software as a Service provides applications for end-users. This is the original service model and it is still the most used model. Examples of SaaS applications are Google Apps and Dropbox.

2.1.3 Cloud deployment models

Cloud computing can be utilized for multiple purposes, therefore it is sensible that it can be deployed in different ways. NIST has defined four different deployment models [2]. These models are developed to fulfill different needs, but they also have different costs. Deployment models are called public, private, community, and hybrid.

1. Public cloud environment is cloud environment that is open for everyone to use. Public cloud provider is usually business, but it can also be some academic or government organization. This cloud model is hosted on the premises of the provider. Usually, when referring to cloud computing in literature or spoken language, public clouds are meant. This is also the case in this thesis.

2. Private cloud environment is an environment that provides services exclusively for certain client or clients. Usually, private cloud exists on the premises of its users but it can also exist off-premises. With private cloud some responsibilities are not present with public cloud, for example, purchase of the hardware and the administration of it.

3. Community cloud provides services for a certain community of users. This community consists of users that have some shared interests. A community cloud is hosted and maintained by some entity or entities of the community, a third party, or some mix of them.

4. Final cloud deployment model is called hybrid. Hybrid clouds consist of at least two different cloud deployment models. In hybrid cloud different cloud environments are connected.

2.1.4 Microsoft Azure

As mentioned at the start of this section, there exist several cloud platform providers, for example Amazon Web Services, Google Cloud Platform, and Microsoft Azure. Case study of this thesis uses Microsoft Azure so it is introduced here shortly. Also, it is introduced, since it is an example of cloud platform.

Microsoft Azure is a cloud platform by Microsoft. According to Microsoft, Azure has more than 200 products and cloud services [4]. These include services like AI, machine learning, IoT, blockchain, virtual machine, mixed reality, web applications, containers, databases, and so on [5]. So, Azure has a really wide supply of cloud services and a portfolio of clients too. According to Microsoft, 95 percent of Fortune 500 companies use Azure services [4].

Microsoft offers a tool called Azure Portal for managing cloud applications in Azure. Azure Portal is a web-based application that can be used with a browser. Azure Portal allows users to do things like build, manage, and monitor for all different applications that are deployed to Azure cloud. [6]

When deploying more traditional web applications to Azure, interesting services are for example, App Service and Azure SQL. Azure App Service is a cloud service for hosting web applications [7]. App Service supports multiple languages and can be run on Windows and Linux based environments [7]. For App Service, one can create WebJobs [8]. WebJobs are back-end tasks related to the App Service application. WebJobs are run in the background and can be monitored via Azure Portal.

Azure SQL provides a couple of products that use SQL Server database engine [9]. This means that these products are almost entirely compatible with existing SQL Server databases, that are used for example on-premises. App Service can also be connected to Azure SQL Server Database. This is done by using user-defined connection string [10]. This way the web app and the WebJobs in the App Service both have access to cloud database in Azure.

2.2 On-premises environment

On-premises web-system is the traditional way to implement web-systems. This means that a company buys an application, which is then installed on servers that the company owns. The company must also buy the servers and the knowledge to install and maintain them. [11]

As mentioned in previous chapter, deploying applications to cloud has become the standard way to implement web-based systems. Before the era of cloud, the standard way for deploying web-based systems was to deploy them on-premises. When considering pre-cloud web-systems, the assumption is that they are working on an on-premises environment.

When looking back to the year 1995, a speech by Professor Sa'ad Medhat was given on international seminar, arranged by IBM [12]. Medhat discusses distributed systems and thin clients. Medhat mentions that in the future as personal computers get more popular, some parts of that computation should be moved from server to client.

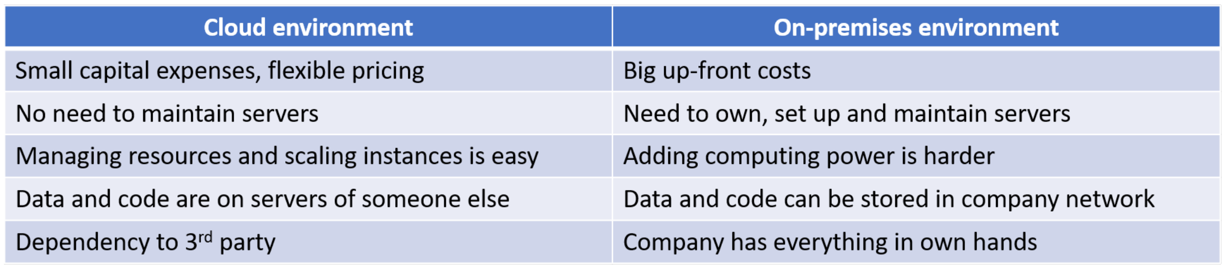
Saetent et al. discuss thin client development in their article [13]. They define thin client as a client that does almost nothing but shows information provided by resource server. This server handles all computation and then serves this content to the thin client. Thin client and resource server communicate with each other within the same network.

In this thin client-server model, most of the computation was handled by a single centralized host computer. It is obvious that this is far from ideal. As Medhat suggested, more computation was moved to the client-side of the application as technology allowed it. As the client-side got even more duties, a new concept of thick client emerged.

2.3 Cloud environment versus on-premises environment

For this comparison, two different sources have been studied. These sources have quite similar analysis on this matter, which makes sense. Differences between cloud and on-premises should be similar, no matter who makes the comparison.

First article is from S. Bibi et al. [14]. They discuss differences between cloud and on-premises environments in their article about whether to acquire on-premises or SaaS-based solutions. This article is inspired by the trend to migrate applications to cloud environments. They also provide a SWOT-analysis of cloud versus on-premises. The second source is from Rabetski and Schneider [15]. They have written an experience report about migrating an on-premises application to cloud. In this report, they also recognize the benefits and problems of on-premises deployments. Figure 2.2 shows the main differences identified in these studies.



Cloud environment	On-premises environment
Small capital expenses, flexible pricing	Big up-front costs
No need to maintain servers	Need to own, set up and maintain servers
Managing resources and scaling instances is easy	Adding computing power is harder
Data and code are on servers of someone else	Data and code can be stored in company network
Dependency to 3rd party	Company has everything in own hands

Figure 2.2: Differences in cloud environment and on-premises environment presented shortly

One big strength of cloud environment, which is mentioned in both sources, is small capital expenses. When going with the on-premises environment there is a big up-front payment, since the company must buy the servers and the software. The company must also have the technical knowledge to run and maintain these servers. With cloud environment, there is no need for additional hardware. Also, cloud environments have flexible pricing, like pay-per-use. Pay-per-use is especially useful pricing method when the application is not needed on a daily basis. [14] [15]

Managing resources and scale of instances in cloud environment is something that is significantly easier than in an on-premises environment. In [14] it is mentioned as strength of cloud environment. This means that it is easy to scale instances horizontally and vertically. Horizontal scalability means the number of instances and vertical scalability means the size of those instances.

Deploying applications on-premises offers some benefits too. One being that code and data of the company are physically close, unlike with cloud. If the code or data, or both are sensitive in nature this can be important. With an on-premises system, it is possible that the data never leaves the network of the organization. [15]

Both sources also mention that cloud applications suffer from latency issues. These sources are from the early 2010s, so this issue might not be that relevant anymore in 2020. In fact, a study published in 2019 does not have full consent of this. Pelle et al. write about latency-sensitive cloud-native applications [16]. They come to a conclusion, that when selecting cloud services carefully, it is possible to use cloud in latency-sensitive applications too.

3 Cloud migration

In this chapter, there is discussion about what cloud migration is and why it should be done. After that, changes in architecture are discussed. The last part of this chapter addresses migration process. It goes through what parts and what tasks the migration process contains.

Before starting cloud migration, companies face decision-making process. Whether they should migrate their application to cloud or not. There is a lot of academic research on this subject that should help with that decision. Assumption in this thesis is that this decision has already been made. Therefore, the decision-making process is not addressed at all. Excluding the decision-making process, this chapter should give a thorough understanding of cloud migrations.

3.1 Migration

When considering cloud migration, it is important to understand how it is done. What does the process of migrating an application to cloud contain and how does it change the architecture of the application? However, before thinking about how to do it, it is important to understand what cloud migration is and why it should be done. In this section, the "what" and the "why" are discussed.

3.1.1 What is cloud migration

In their study Al-Azzoni et al. discuss performance of the software migration [17]. Migration is said to be the activity where parts of a legacy application are migrated to a more modern platform. One or more parts can be migrated from the legacy application. Parts to migrate can contain hardware, operating system, source code, languages, and databases. This study describes software migration in general, but the same definition can be used with cloud migration too.

Müller explains re-engineering strategies for software migrations in his study [18]. This study identifies what challenges there are in software migrations. These challenges are found to be the scale of the legacy application, corruption of the legacy application, outdated documentation, and implementation technologies that have aged.

There are a lot of legacy applications that could benefit from migrating to cloud environment. Often these legacy applications are hosted in on-premises environments. While cloud computing is rapidly growing and evolving, it is becoming even more interesting topic for a lot of researchers and companies to figure out what it would take to migrate these legacy applications to cloud environments.

3.1.2 Why should cloud migration be done

Before making the first step to migrate the application to cloud environment, companies face a decision-making process whether the migration should be done at all. Research field has a lot of support for making this decision. In 2013 was made systematic review [19] on existing cloud migration researches, and it was found, that majority of cloud migration-related researches focus specifically on this topic. To be exact, 14 out of 23 (61 %) reviewed researches focused on decision making.

In a study about why companies migrate to cloud [20], it is found that there are a couple of different motivating factors for companies to migrate their legacy applications. Factors that are found in literature are IT costs, flexibility and scalability of the application, faster time-to-user, and reducing IT complexity. The study also contains two case studies, that both have one motivating factor, that literature did not mention. This is the need to improve or innovate business processes.

Marston et al. have identified the strengths, weaknesses, opportunities, and threats of cloud computing in their paper about the business perspective of cloud computing [21]. For this paper, they have interviewed multiple executives that have a deep understanding of this subject. For key advantages, they have listed four different things.

1. It makes it a lot easier for small companies to gain access to compute-expensive processes. These processes are often a rather short time only and as cloud services have pay-per-use politics, it is more suitable for smaller companies too. Cloud computing can be also useful in developing countries that have fallen behind in the IT revolution. Cloud computing provides computing power to places that would have otherwise lacked it.

2. Access to hardware resources is immediate and demands no up-front investments. Unlike on-premises environment that demands those up-front investments to get things running.

3. With cloud computing, services can be scaled easily. When there is demand for more computing power, it can be deployed fast and easily. Resources can also be viewed accurately through systems that cloud provider offers. An example of such system is Azure Portal, that is introduced in 2.1.4.

4. With cloud computing it is possible to create different types of applications that were not previously possible. For example, with internet access it is possible to read emails and access personal files on any computer.

3.2 Architecture

Migration of legacy application can be done in different ways since it is not always desired to migrate the entire application stack as-is. Term *application stack* refers to common three-tier architecture, that consists of presentation layer, business layer, and data layer. Migration types can be sorted in different ways. However, when considering different migration strategies, five R's or six R's of migration strategies are often mentioned. Six R's are presented to answer RQ2.

3.2.1 Six R's of migration strategies

Five R's of migration strategies were first introduced by research company Gartner in 2011. After that they have appeared in multiple academic researches, usually as six R's of migration strategies[1]. Names of those six R's might change a bit in different sources, but the content stays more or less the same. Ahmad et al. (2018) [23] define these six R's to be Rehost, Replatform, Repurchase, Refactor, Retain, and Retire. Figure 3.1 shows these six R's in short.

Rehost is also called "Lift-and-shift", which means migration that is done on IaaS-level. In Rehost, the whole application is migrated from non-cloud environment to cloud environment. This migration type doesn't require changes to the architecture of the application. Rehosting can be carried through with little or no reconfiguration. Since the entire application is only rehosted, this migration strategy is relatively fast and easy. [22]

In *Replatform* strategy, one or more of the three layers of the application stack are converted to use PaaS offerings. Small changes to architecture are possible, to fully utilize cloud environment. This migration process might contain some optimizations to code and other configurations to fully utilize the features of the cloud. When considering amount of work in migration, replatforming is middle ground for rehosting and refactoring. [23]

[1]Sources, where six or seven R's are mentioned: [22], [23], [24]. These strategies are also identified in many unacademic sources, such as Forbes [25] and Amazon [26]

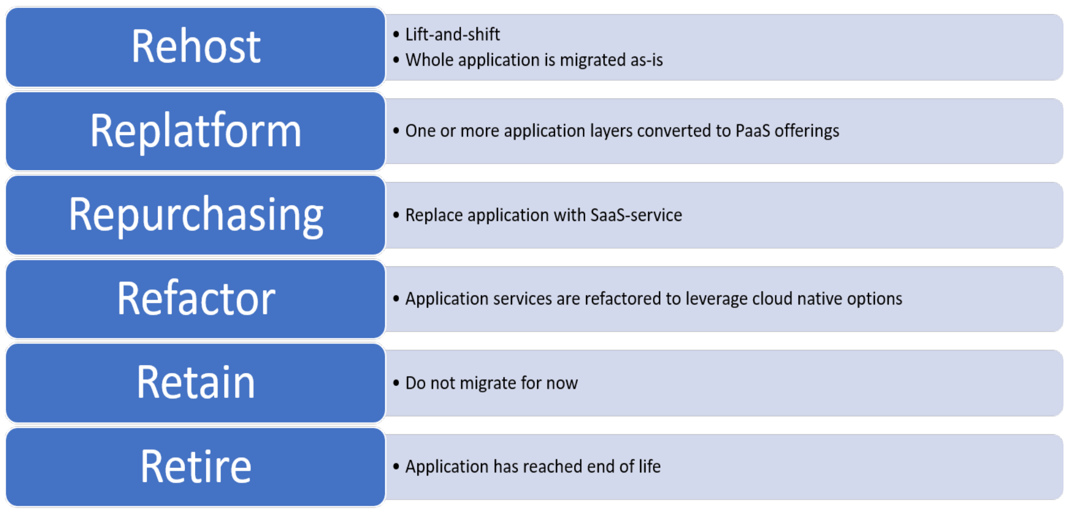

Rehost
• Lift-and-shift
• Whole application is migrated as-is
Replatform
• One or more application layers converted to PaaS offerings
Repurchasing
• Replace application with SaaS-service
Refactor
• Application services are refactored to leverage cloud native options
Retain
• Do not migrate for now
Retire
• Application has reached end of life

Figure 3.1: Six R's

Repurchasing means replacing the application with SaaS-service. The new application should replace the legacy application and contain similar features and characteristics that can be found from the legacy application. Preferably with improvements, of course. One example of such migration would be changing complex customer relationship management platforms to Salesforce. [23] [22]

Refactor means migration where services of the application are refactored to leverage cloud-native options. To fully leverage cloud-native options there are a lot of changes that need to be done. Architecture, codebase, and even database queries might need some changes in this migration strategy. Motivation for this strategy is having special needs for the application such as need for new features, more performance, or other up-scaling. Unlike rehost-strategy, this migration takes a lot of work. [22] [24]

Retain is a good strategy when migration of legacy application would cause too much costs, maintenance, or troubles for current users. Of course, migration should not be done if it does not make sense. When this is the case, migration can be refrained and maybe revisited later to see if migration has come topical. [23] [22]

Last strategy is *Retire*. It is found that companies have applications and services that are no longer in use and therefore are not beneficial. These applications should be retired.

[23]

3.3 Migration process

For cloud migration, there exist cloud migration frameworks that provide basic steps that should be done to achieve successful migration. For this thesis, four models have been examined and their similarities have been considered. After finding these similarities, three phases have been defined. These phases are called Planning, Migration, and Deployment.

Jamshidi et al. (2013) wrote a literature study that compared existing studies about cloud migrations. With this information, they created their own model for cloud migration. This model is called cloud reference migration model (CRMM). [19]

Second model is called REMICS (Mohagheghi et al. 2010). It is short for Reuse and Migration of legacy systems to Interoperable Cloud Services. REMICS model provides a model-driven methodology for migrating legacy systems to cloud. With multiple different models describing different phases, it is possible to create an exact description of all phases. Accuracy comes with a price. Since this is model-driven, it is needed to go through multiple different models, in addition to the actual migration model in order to use the migration model. REMICS project has been funded by the European Commission. [27]

Third model is called CloudMIG. CloudMIG is created by S. Frey and W. Hasselbring in 2011. CloudMIG is also a model-driven methodology for cloud migrations. Similar to the REMICS-model, CloudMIG also demands users to understand multiple models to use the actual migration model. Main goal for CloudMIG is to support the migration of legacy on-premises applications to Paas and Iaas applications. [28]

Last model is not a model but a metamodel for creating migration models. Study (Panami et al. 2019) goes through existing cloud migration literature and then introduces metamodel based on that literature. Goal for migration metamodel is to be helpful when creating case-specific migration models. [29]

None of these four models consider what happens when application is in maintenance phase. Because maintenance is an important part of lifecycle of an application, it is added as fourth phase to this thesis too. First three phases (planning, migration, deployment) have been written using these four models but the maintenance phase has its own sources.

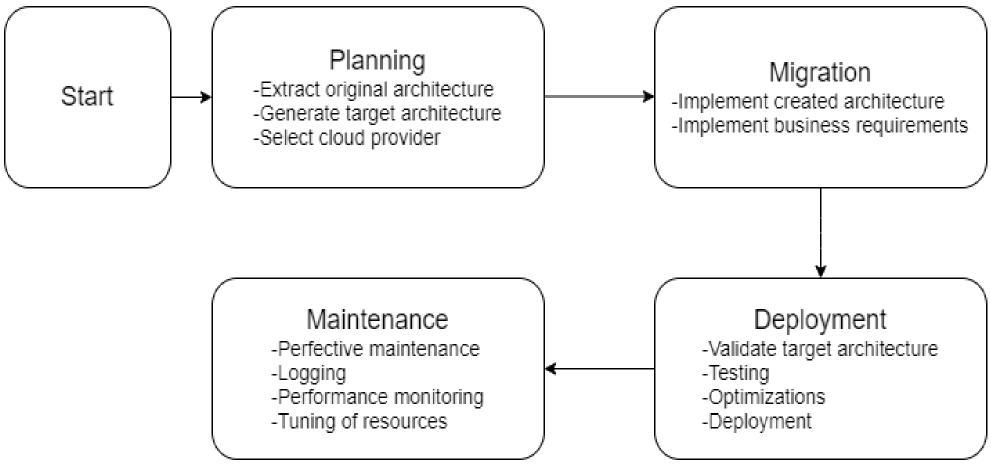


Figure 3.2: Migration process

In Figure 3.2, all phases of the process are presented in short. This image shows each phase and the biggest tasks inside them.

3.3.1 Planning

Planning phase should be done to fully understand what the goal of this migration is and where does the application stand now. In planning phase, there are three different main activities that should be done. Each of these main activities contain smaller tasks.

First main activity is extracting the original architecture of the legacy application [27]–[29]. It is known that the architecture of the application tends to corrode over time [28]. This can be caused by, for example, sudden changes in requirements or features. Therefore, the original architecture might not be there anymore.

To extract architecture from corroded application it is needed to analyze at least source code, dependencies, documentation, and knowledge of users and developers. Analyzing these things gives us findings such as knowledge about components in the application, overall implementation, and business requirements. Business requirements are something that should be explicitly listed so that they can be used later in the migration phase. [27]

These findings are useful when we try to see if there are some technical requirements, such as hardware requirements. They can also be used when selecting cloud migration strategy. Because of this, the mentioned findings should be recognized, even if the old architecture would not be analyzed.

Second main activity is generating target architecture [19], [27]–[29]. It is needed to decide cloud migration strategy and then design new architecture after that. Decided cloud migration strategy will most likely determine a big part of the target architecture. Whether the strategy is to rehost (“lift-and-shift”) or refactoring, it will determine a big part of the target architecture.

When considering target architecture, it is good to notice that in some cases it might be beneficial to add or remove components when migrating to cloud [29]. Target architecture should be created in a way that the application could maximize the benefits of cloud environment. All components might not be suitable for cloud environment performance-wise or they might be completely incompatible.

Third main phase is selecting a provider for the cloud environment [19], [29]. The provider needs to have services that suit the migration. When selecting a provider, it is needed to go through issues related to licensing and other things that are specific to a certain provider.

3.3.2 Migration

Second phase also consists of two main activities, which contain smaller tasks. This is similar to the planning phase. Goal of the migration phase is to migrate all identified business requirements successfully from old to new architecture.

First main activity of this phase is to build the architecture that was generated in the planning phase [19], [27]–[29]. Building the architecture is done by applying case-specific methods and maybe even by replacing components with cloud offerings. In migration phase, it is possibly needed to add or remove components too.

Building the new architecture often contains adapting the codebase to suite new architecture and cloud environment [19], [29]. Also, when architecture changes, there is usually a need for reconfigurations. And to know how to do reconfigurations, it is needed to identify which cloud resources will be used.

Second main activity in migration phase is to fully implement business requirements that were identified in the planning phase [27]. While doing this, there's also a need to validate that these requirements are met and that the new application still works as intended.

Also, often when migrating an application, new requirements for the application have emerged. This means that while doing a migration, possible new features should be also implemented.

3.3.3 Deployment

Like the first two phases, deployment phase also contains main activities that have subtasks. There are four main tasks in this phase, that are validating target architecture, testing, optimizing services for the cloud, and deployment. The first three tasks are preparation tasks for the deployment.

Validating target cloud architecture is done with the help of target architecture that was created in the planning phase. To be able to validate created cloud architecture, it is needed to define validation criteria. Validation criteria should be created from the target architecture that was created in the planning phase. [19], [27]–[29]

Second main activity is testing. Before actual testing can start, it is needed to create a test environment. The test environment should be similar to the actual deployment environment. The application should then be divided into smaller modules that can be tested separately. After finding these modules, it is time to do the actual testing. Test cases should be such that they can be run on old and new system [27]. This way it is easier to find the differences between these two systems. [19], [27], [29]

After running tests, the next main activity is to do optimizations for the new system. Since the old system is an on-premises system and the actual migration work is most likely done locally there might emerge some optimization needs for the cloud environment. [29]

Final activity of deployment phase is the deployment of the application. Deployment contains different elements, depending on the cloud provider and migration strategy that was chosen in the planning phase. Also, configurations are often necessary in the deployment phase. [19]

3.3.4 Maintenance

Planning, migration, and deployment were phases that were written to this thesis using earlier mentioned four migration models. However, these models did not contain tasks that are related to the maintenance of the application. It is known that maintenance costs are often notable part of total costs in software project [30]. Therefore, maintenance is also added to this process, even though original models do not have it. This phase is a compilation of different sources. To make this phase thorough, it was first wanted to find out what regular software maintenance contains. After that, it was wanted to find out what maintenance tasks come with cloud environment.

In a publish about maintaining traditional software applications, there are two types of maintaining works found. These same maintaining types apply for cloud applications. First type is called perfective maintenance. This type contains work, where functionalities of the system are enhanced. [30]

When doing enhancements to some application, it might have an impact on other applications and components that are linked to it too. This leads to more maintenance work, where you update all linked components to keep them consistent with each other.

[30]

In a study about logging framework for cloud applications, it is said that logs are one of the most important pieces of analytical data in cloud applications. This study finds multiple use cases where application logging is beneficial: debugging, fault monitoring, troubleshooting, performance monitoring, and so on. [31]

To enable logging, this study suggests three steps that need to be done. First, enable logging on all different application components. Next, setup log transport. This means that logs need to be transferred from the place they are logged to some central place, where they are easily accessible. Final step is to tune logging. If possible, log messages should tell what happened, when it happened, who triggered it, and why it happened. [31]

Study about the Quality of Service based cloud application management finds different things that should also be regularly managed in cloud applications. These things are continuous monitoring of application services and tuning of cloud applications under different requirements and budget constraints. For example, in Azure environment, monitoring and tuning can be done via Azure Portal. Azure Portal was introduced in chapter 2.1.4. [32]

4 Case description

4.1 About Aveso

Aveso Oy is an IT consulting company that is specialized in data quality, master data management, internet of things and data warehousing solutions. Aveso was founded in 2014 and it has currently 16 employees. Aveso is Microsoft Analytics Gold partner and it has developed several products that are used in telecom, manufacturing, and energy industries. Two Aveso applications are relevant for this thesis, Aveso Data Quality Tool and Aveso DataHub.

4.1.1 Aveso Data Quality Tool

Aveso has developed application called Data Quality Tool for continuous data quality management. Aveso Data Quality Tool is an on-premises web-based application, that offers centralized solution for organizations to manage their data quality process.

Aveso Data Quality Tool was created to support centralized data quality process. Traditionally data quality has been controlled via system controls, user instructions and data quality reporting. This has produced little or no quantitative data that could be used to measure overall data quality in an organization. With Aveso Data Quality Tool, it is possible to create automated data quality controls and, through time, quantitative metrics to assess the data quality in a specific environment.

Today, data is considered to be an important asset for companies. Business decisions should be made based on reliable data to improve certainty. Bad data costs money for companies through manual fixing work, reclamations, and loss of revenue. Digitalization, AI and robotics require reliable data.

Typical data quality issues are caused by user typos, lack of user instructions, changes in data quality requirements, and bugs in system integrations. With Aveso Data Quality Tool it is possible to create data quality rules that expose data quality issues. These SQL based rules define qualification criteria for data objects. It is also possible to create hierarchical rule structures. After creating such rule or structure, it can be executed and a result data set is created. These executions can be scheduled, and they can be recurring for example daily or weekly. Results show error statistics and the actual erroneous data. Aveso Data Quality Tool has integrated Microsoft Power BI reporting. Via this reporting tool, it is possible to follow data quality trends for example per system, data domain or system object. These reports show total number of errors, number of errors within some time period, and number of errors fixed within time period.

4.1.2 Aveso DataHub

DataHub is the second application from Aveso that is introduced in this thesis. DataHub is an application that is created for data collection, master data management, to give reference data, and to correct erroneous data.

Collecting data in ERP-systems is well defined process, unlike collecting data before entering it to ERP-system. There are often excel-files, emails, or other bad ways to gather data before it is entered to ERP. When handling data with these methods, some problems emerge, such as version controlling. These are the problems that DataHub is for.

Aveso DataHub supports data model definition and provides features for managing and editing data before it is inserted to ERP-system. It is a way to collect all pre-ERP data to one place for easier management. DataHub is an Microsoft Azure cloud application, so it can be easily provided for external users, such as suppliers and subcontractors. DataHub also has the ability to export data into Excel, which can then be edited and imported back into DataHub. DataHub workflow is presented in Figure 4.1.

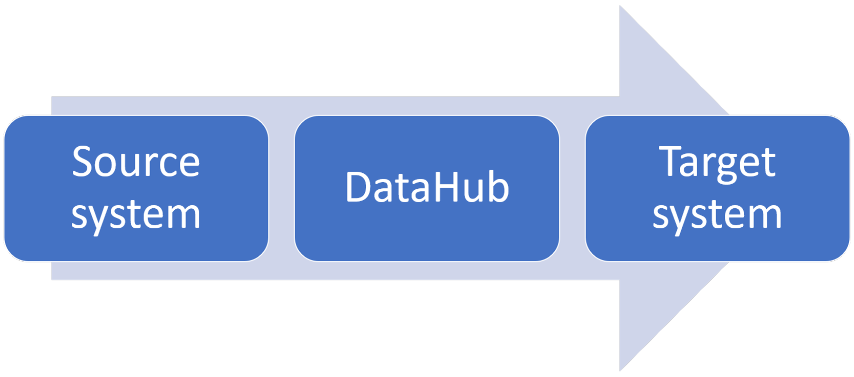


Figure 4.1: DataHub process

4.2 Research method

The object of this thesis is to create a general model for migrating legacy applications from on-premises to cloud environments. First task was to find out from literature what it takes to do such migration. This was done using literature. After that, it was time to generalize this information and form a model that could be followed when migrating legacy applications. Next step is to compare this model with what was done in case study. This comparison gives us insights about the differences between created model and an example project.

Another interesting topic for this thesis is to find out what happens to application architecture when migrating it to cloud environment. Literature provides some architectural models to describe this change. These are described in chapter 3. Literature based architectural models are compared with what happened in case migration. Architecture of case migration is presented in chapter 5 and it is compared with literature in chapter 6.

For these mentioned comparisons it is necessary to get full understanding of how the case study was conducted. This understanding will be gained through interviews. The migration team is rather small, with only seven people including me. Therefore, the planned interview process is somewhat unconventional.

Suitable interviewees are selected based on the goal of these interviews. As mentioned earlier, goals are to find out what was done in the migration project and what were the architectural changes. This narrows number of suitable interviewees from seven to four. These four interviewees had different tasks in the project. There were two developers, a consultant, and a product owner.

First there will be four individual interviews. Goal for individual interviews is to find out how the migration process went in case study. Each interviewee will answer the same questions. Questions are about the migration process, what was done in the process, and what was gained from the process. When doing the interviews, the interviews will be recorded, and later a summary of each interview will be written. These interviews will be compared with each other. After comparison, it should be possible to create a timeline about the process that contains all the things that were part of case migration.

After individual interviews, a group interview should be kept for all interviewees. There is uncertainty whether only four individual interviews could give thorough answers about the migration. Therefore, the individual interviews should be verified and possibly supplemented with a group interview. In group interview, interviewees should see outlines of their combined answers. They can then discuss and analyse whether the created process is insufficient or otherwise lacking something. The goal of the group interview is to verify the timeline that will be created after individual interviews. After the group interview, there should be clean and verified image of what was done in the case migration and what were the architectural changes in it. Figure 4.2 summarizes the interview research that will be used.

It is important to do individual interviews before the group interview since the interviewees have really different backgrounds. For example, one interviewee is a developer that has been working for five months while another interviewee is product owner. This difference would make it really hard to get honest opinions from the developer in the group interview. Because of this, it is decided to first ask questions from everyone individually and only after that do the group interview.

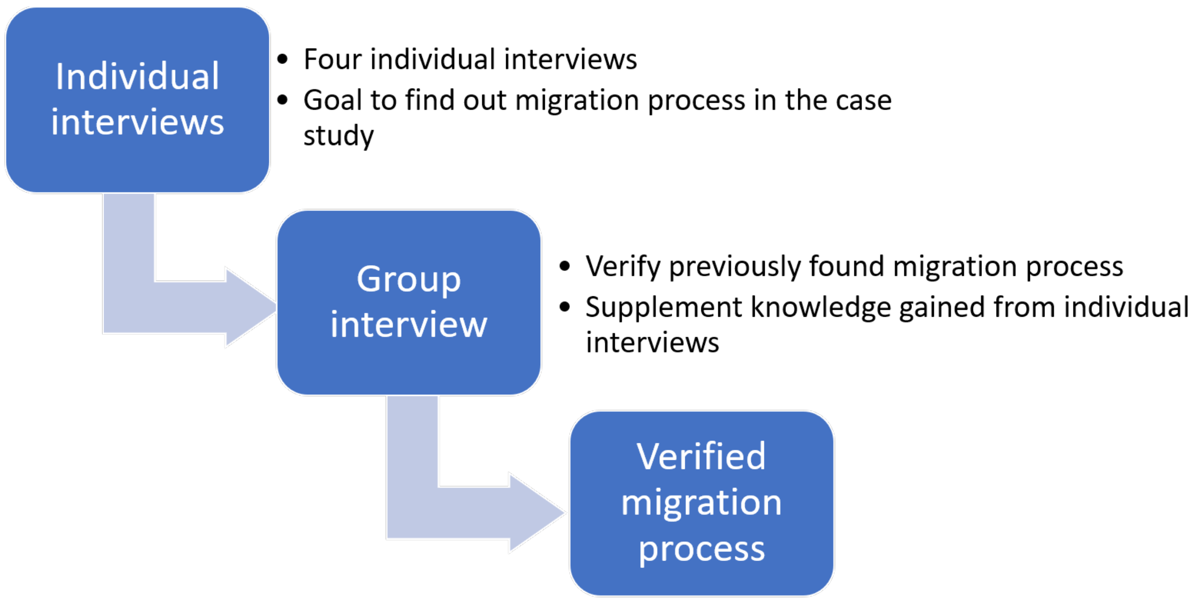

Individual interviews
• Four individual interviews
• Goal to find out migration process in the case study
Group interview
• Verify previously found migration process
• Supplement knowledge gained from individual interviews
Verified migration process

Figure 4.2: Method for finding migration process in case migration

Another way to conduct this research would have been to do only group interviews. Creating a couple of homogenous groups of three to five people and let each group discuss the project could have been a good method too. Unfortunately, the project was not that big, and it only had seven people working on it. Only four of those seven were suitable for answering the questions, which meant that group interview method could not be used.

At the time of the interviews, the case migration was almost done. There was nothing left to do but some cleaning up and final bug fixes. When this thesis was started, the migration had already passed halfway point of the project. Results of interviews are explained in chapter 5. Research method described in this section and in Figure 4.2 was followed precisely when conducting interviews.

If the case had not been started before creating migration process model in this thesis, it would have been sensible to do the case migration using that model. However, since the case had already gone a long way that was not possible anymore. Therefore, it was decided that the case migration could be used as a validation for the migration model created in this thesis.

5 Case study: migrating Data Quality Tool

For this thesis, it was needed to verify the created process model with some case. Aveso migrated their Data Quality Tool to cloud environment and that migration is also the case study in this thesis. This migration is somewhat uncommon since the application was not only migrated to cloud, but also merged with Aveso DataHub.

When these two applications were merged as one, they were given a new name, DataPlatform. This name is somewhat problematic since Microsoft has a product with the same name. Therefore, DataPlatform of Aveso is most likely going to have another name at some point. When writing this thesis, DataPlatform was not quite ready yet and had no paying customers.

Since DataHub was already working on Azure cloud it made sense to keep it as a base project. All references to the name Datahub were changed to DataPlatform in that project. After renaming, it was time to start merging features from Aveso Data Quality Tool to that project.

The idea of this chapter is to depict what has been done in this case. All mentioned things are found through interviews that were held during the research phase. Research method is explained more closely in 4.2.

5.1 Case process

Migration of Aveso Data Quality Tool can be presented in four phases, similar to the previously created model. These phases are planning, migration, deployment, and maintenance. Content of phases has been examined with interview research that was presented in section 4.2.

5.1.1 Planning

In the planning phase, there were different things that were done. Project participants had meetings where they discussed current state, technical problems and future requirements. After these meetings individuals then added details to the bigger picture.

In meetings, there was discussion about the current state of the application. The discussion based on thorough knowledge of participants and documentation. Functions, technical implementation, and architectural decisions of the application were revised. With this revision, it was supposed to find possible technical limitations that may occur when migrating from on-premises to cloud. In their migration model [29] Panami et al. also state that it is important to identify possible incompatibilities early.

Some technical problems and limitations were found. For example, Windows authentication can not be used, when the application is deployed to Azure. When going through the documentation of Microsoft, it mentions that "You can use Windows authentication when your IIS server runs on a corporate network that is using Microsoft Active Directory service domain identities...". [33]

After finding these problems, it was needed to bypass them somehow. For example, since Windows authentication did not work, the application started using a third party component called IdentityServer. With IdentityServer it was possible to add Azure authentication and local users for the application.

In addition to considering the current state of the application, the meetings contained discussion about future requirements. Aveso has clients who are using Aveso Data Quality Tool or Aveso DataHub and they have implied their interest in an application that is a combination of these two. To make a satisfying application that contains functionalities from both, it was necessary to discuss the future too.

This application had one person who was responsible for the new architecture. Meetings had discussion about the current state, technical challenges, and future requirements of the application. Based on these meetings, the architect created new architecture for the application.

The new architecture had more components than the original architecture. This was done because of technical difficulties and to utilize characteristics of cloud. For example, the old application had some SQL executions that could be scheduled and executed by SQL Server. These functionalities were detached from SQL Server and new components were made for both.

In the planning phase was also a concern about the scope of the migration work. To keep this scope as small as possible, it was decided that the migration should be done with as few changes as possible. This meant that all the functionalities should be migrated as-is and possible changes to them should be done afterward.

Also, in the early planning phase, some preliminary estimate for workload was made. This first estimate was there to only give guidelines. The idea was to specify the workload estimate later as the project goes further.

5.1.2 Migration

In this project, Azure DevOps was used for development, project management, and collaboration. With Azure DevOps, it is possible to list project work items and to track them in Kanban style. Each work item can be marked as to do, doing, or done. There are a lot of customization possibilities with Azure DevOps but for this thesis, it is not relevant to go through these possibilities.

At the beginning of migration work, it was decided that work items could be divided into different abstraction levels. Abstraction levels were decided as follows: Epic, Issue (or Bug), and Task. These levels are depicted in figure 5.1.

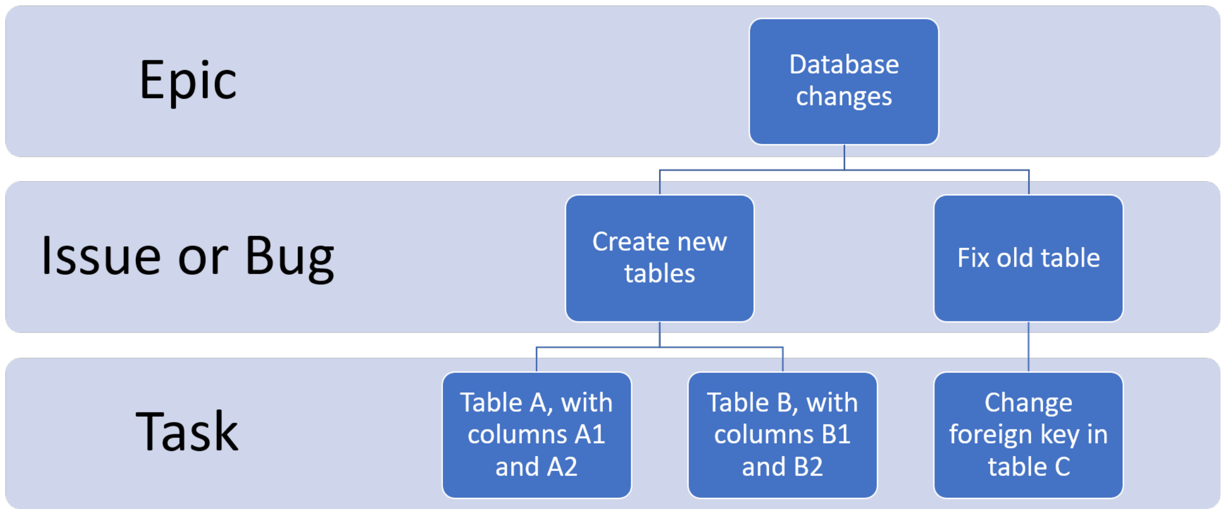

Epic
Database changes
Issue or Bug
Create new tables
Fix old table
Task
Table A, with columns A1 and A2
Table B, with columns B1 and B2
Change foreign key in table C

Figure 5.1: Example of work items

After making this division for work items, it was time to write down all Epics. Epics were work items that were identified in the planning phase. When it was time to start implementing some Epic, that Epic was specified all the way down to Task level.

Since the front-end of Aveso Data Quality Tool was developed with a different web-application framework than the front-end of Aveso DataHub, it was needed to rewrite all code in the front-end of Aveso Data Quality Tool. Rewriting front-end code gave a chance to write the code to equate to modern best practices and standards. The front-end code of Aveso DataHub was also refactored to equate these best practices and standards. Figure 5.2 shows how these two applications were merged.

In on-premises environment, Aveso Data Quality Tool was implemented with software development framework called Angular and DataHub with framework called React. As the target was to combine these applications, Data Quality Tool was rewritten with React. So, the change of the web-application framework was only done to unify it with the framework that was used in DataHub. This was not a necessary step regarding cloud migration.

Both applications used the same technology for implementing their APIs. This made it possible to just copy and paste all API related code from Data Quality Tool to DataPlatform. Before copying the Data Quality Tool code to DataPlatform, the DataPlatform API was updated. All code that was copied to DataPlatform after that might have been subject to small or big changes since the base was updated.

Moving Data Quality Tool API to cloud made no difference in how the API works. Because of this, all the changes that were done to Data Quality Tool API were done only because of new features of the updated technique. Not because of migration to cloud environment.

Both applications also used the same database, which is SQL Server. Almost all features of SQL Server that work on-premises work on Azure SQL service. Because of this, the need for changes in the database of Aveso Data Quality Tool was small.

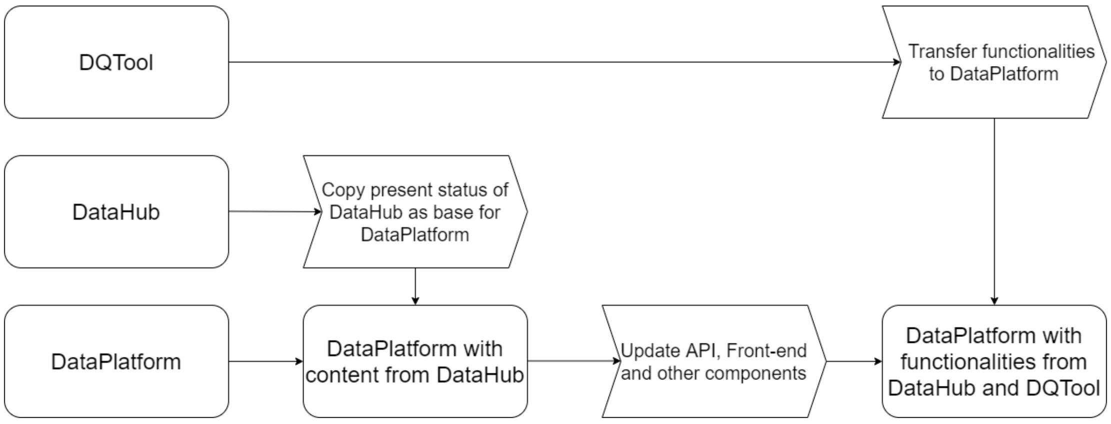

DQTool
Transfer functionalities to DataPlatform
DataHub
Copy present status of DataHub as base for DataPlatform
DataPlatform
DataPlatform with content from DataHub
Update API, Front-end and other components
DataPlatform with functionalities from DataHub and DQTool

Figure 5.2: Merging two applications in migration phase

To be able to analyze what is happening in the application, it was needed to add logging for the application. Logging was added to all application components for this purpose. All application logs can be accessed through Azure Portal.

One goal in the migration phase was to fully implement the new architecture, that was created in the planning phase. New architecture contains components that were not part of Aveso Data Quality Tool or DataHub originally. These components had to be created from the beginning.

When developers had some new features ready, they tested it by themselves first. After that, new features were introduced in weekly meetings. In these meetings, other project participants had a chance to comment on new features. Finally, when some feature was thought to be ready it was time for the product owner to test it.

In weekly meetings, Azure DevOps was reviewed every time to see how the project is going. By following lists, in DevOps, it was possible to make sure that architecture and all functionalities were implemented as planned.

5.1.3 Deployment

When the application was ready to be deployed, it was time to create a test environment. Test environment had to be similar to the environment that the application would be deployed on customer cases. Therefore, an App Service plan, SQL Server, and WebJobs for App Service plan were created to Azure.

Aveso Data Quality Tool has a list of everything that is possible to do in the application. This list was updated to equate functionalities that are in DataPlatform. After the test environment was deployed to Azure, the product owner started to go through this updated list. When going through this list, it was easy to find all bugs, since the list is thorough. Also, if something was missing from the application, the list revealed that too.

When deploying the application for a customer, some implementation tasks are needed. First, the application needs to be connected to the data of the customer. Main idea of the application is to help with data quality and to pre-process data before entering it into ERP. Therefore, connecting customer data to the application is an essential step. After this, there are still some other configurations that need to be done. For example, connecting all separate architectural components together.

Final step of deployment starts when the application is up and running in the cloud.

Since the application is new, it is possible that all users are not aware of its functionalities. Therefore it might be needed to familiarize customers with the application. This is done with a user manual and personal assisting.

5.1.4 Maintenance

Maintenance phase contains different kinds of tasks. Some tasks are done regularly whereas some tasks are initialized by a customer. Aveso can sell DataPlatform as SaaS or it can be installed into a cloud environment that is owned by the customer. Maintenance tasks are different, depending on how the application is sold and what kind of agreement is done with the customer.

When the application is sold as SaaS, it includes more maintenance work than just selling the application to the environment owned by the customer. The application needs to be monitored regularly. It is needed to see that the application is still up and running and that there are enough resources allocated for it. From this monitoring, you can see if it is needed to scale the capacity up or down. Monitoring state of the application can be done via Azure Portal. Azure portal is explained in section 2.1.4.

It is understandable, that it does not really make sense to keep checking on the application all the time, just to see that it is up and running. Therefore, Aveso will configure alerts to Azure. These alerts can notify you of several things. For example, if the application has stopped or if the usage of CPU or memory has jumped over some threshold. Alert for stopped application is a really useful feature and it will be configured to DataPlatform, when it is sold as SaaS.

Customers can report bugs and improvement ideas for the application. These are listed to Azure DevOps, like in the migration phase. These listed things are considered when doing version updates. DataPlatform will receive version updates regularly. These new versions consist of non-urgent bug fixes, improvement ideas, and other updates, such as updating used technologies. Of course, if a customer reports a bug that is urgent, a hotfix can be sent.

When a customer sends a report of a bug, it will be investigated. When troubleshooting what is wrong, application logs play an important part. There can be error messages or other abnormal content that will help to find out what is wrong. As R.Marty mentions in his study [31], logs are one of the most important pieces of analytical data in cloud applications.

As explained in this section, different delivery and agreement types have different maintenance scopes. This is illustrated in Figure 5.3 below. When the application is sold as SaaS, Aveso will take care of the environment of the application as well as the software. However, if the application is sold to cloud environment of the customer, then Aveso will provide only regular software maintenance support.

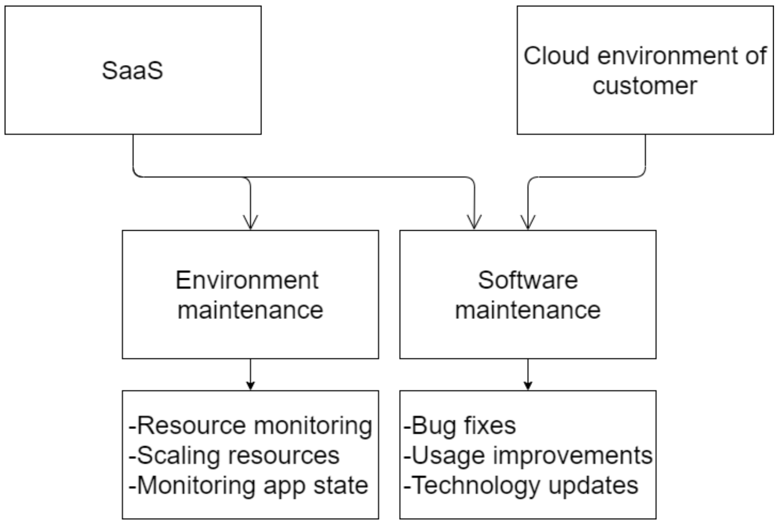


Figure 5.3: Maintenance scope is different for different delivery types.

5.2 Case architecture

5.2.1 Architecture before migration

Aveso Data Quality Tool and its functionalities were introduced in section 4.1.1. Main function for Data Quality Tool is the maintenance of data quality. To maintain data quality it is needed to access customer data. This means that the rest of the system is dependent on that data. In figure 5.4 customer data is depicted as source database.

Before migrating Aveso Data Quality Tool to cloud environment, the architecture had only five components. These components are illustrated in Figure 5.4 below.

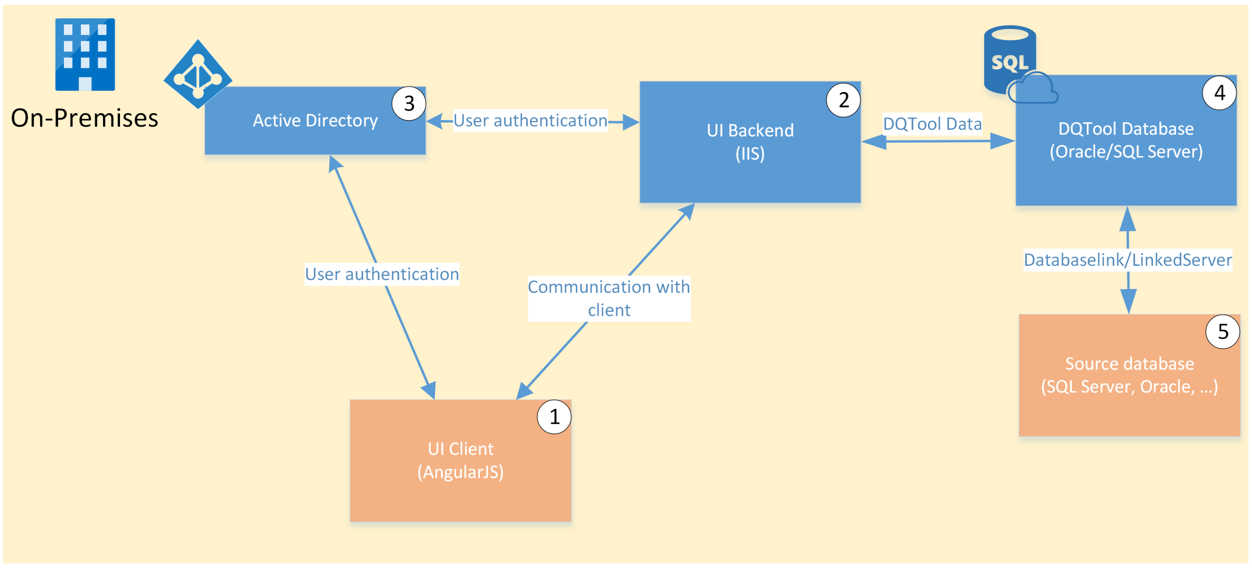
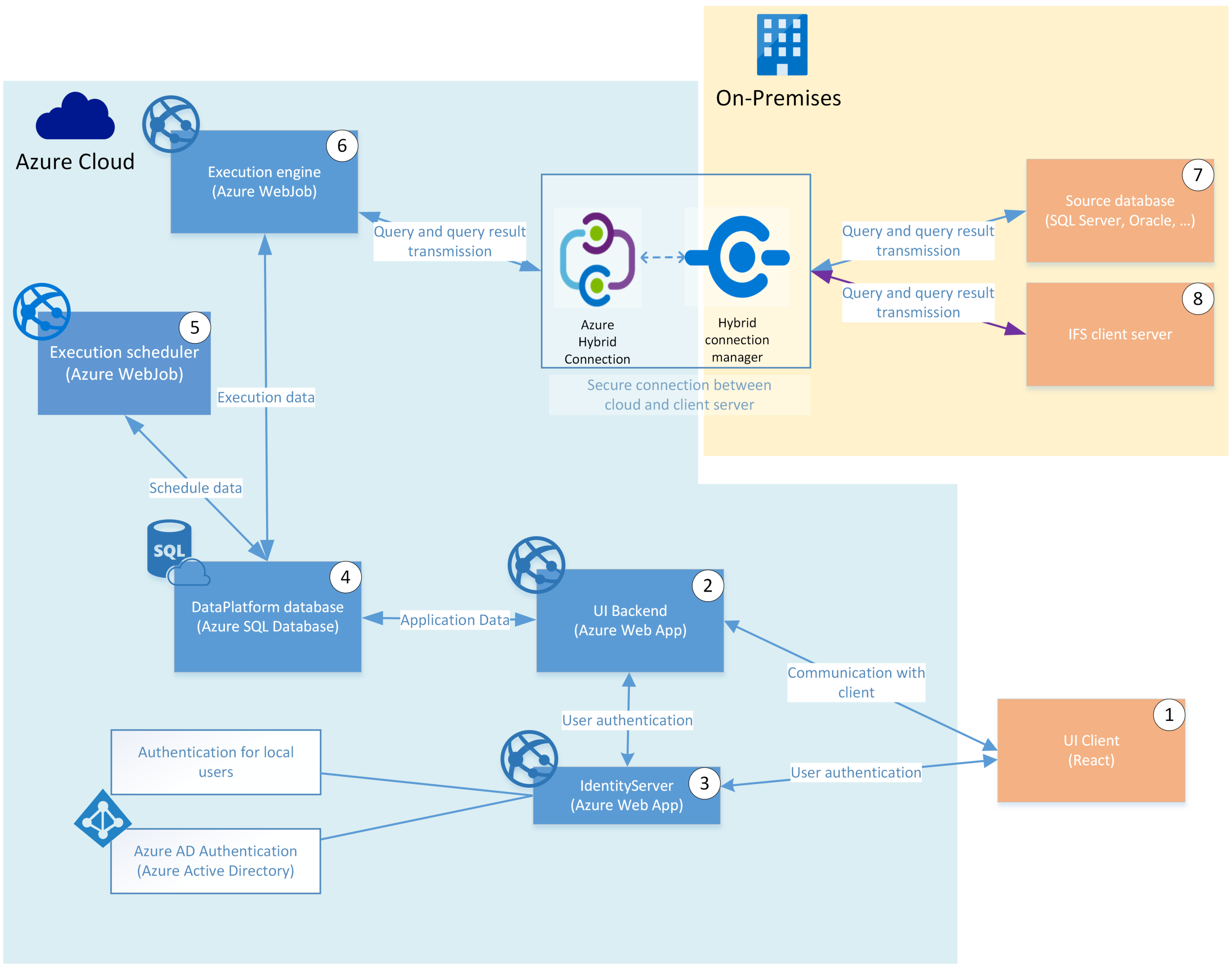

On-Premises
Active Directory
3
User authentication
UI Backend (IIS)
2
DQTool Data
SQL
DQTool Database (Oracle/SQL Server)
4
Databaselink/LinkedServer
Source database (SQL Server, Oracle, ...)
5
User authentication
Communication with client
UI Client (AngularJS)
1

Figure 5.4: Architecture of Data Quality Tool

1. UI Client that runs on web-browser.This is the entrypoint to application.
2. Back-end that runs on servers of a company. Physical server uses software called Internet Information Services to run the application. Back-end communicates with UI Client and databases.
3. Microsoft Active Directory is used by client and back-end to authenticate users.
4. DQTool database is a database that is dedicated to the use of the application.
5. Source database is connected to application via DQTool database. It provides necessary data from the customer to the application.

5.2.2 Architecture after migration

Migrating Aveso Data Quality Tool to cloud environment meant a lot of changes to architecture too. As mentioned in section 5.2.1, the application needs to be connected to the data of the customer. This is the only part of the new architecture that does not have to be in cloud environment. Azure hybrid connection makes it possible to connect DataPlatform to servers of the customer, even if the servers are in on-premises environment. New architecture is illustrated in Figure 5.5 below.



On-Premises
Azure Cloud
6
Execution engine (Azure WebJob)
Query and query result transmission
Azure Hybrid Connection
Hybrid connection manager
Secure connection between cloud and client server
Query and query result transmission
Query and query result transmission
7
Source database (SQL Server, Oracle, ...)
8
IFS client server
5
Execution scheduler (Azure WebJob)
Execution data
Schedule data
SQL
4
DataPlatform database (Azure SQL Database)
Application Data
2
UI Backend (Azure Web App)
Communication with client
User authentication
1
UI Client (React)
Authentication for local users
3
IdentityServer (Azure Web App)
User authentication
Azure AD Authentication (Azure Active Directory)

Figure 5.5: Architecture of DataPlatform

1. UI client works similarly in web-browser as it did before migration.
2. UI back-end of the application is now running in Azure. The purpose of this component is the same as it was in the original application. The difference is that this is now deployed to Azure as App Service plan. App Service plan was introduced in section 2.1.4 and in source [7].
3. As mentioned in section 5.1.1 and in source [33], Windows authentication could not be used in Azure. This was replaced with IdentityServer that provided a way to use Azure authentication and local users for the application. IdentityServer is deployed as a separate App Service plan to Azure. However, it is working under the same URL as the main application.
4. DataPlatform database was called DQTool database in the original architecture. The purpose of this component has stayed the same. It is a database that the application can use for its needs. The difference is that this new component is now Azure SQL Database instead of on-premises Oracle or SQL Server database. Azure SQL Database was introduced in section 2.1.4 and in source [9].
5. Execution scheduler is a new component. It schedules executions that are run by component called Execution engine. Execution scheduler is deployed as a WebJob under App Service plan of UI. WebJobs were introduced in section 2.1.4 and in source [8].
6. Execution engine is also a new component. Like execution scheduler, this is deployed as a WebJob too. Execution results from the execution engine are transported to the source database and ERP of the customer.
7. Source database has the same purpose as it did on the original application. It can also be deployed on-premises, similarly to the original application.
8. IFS ERP also receives data from DataPlatform and sends data to DataPlatform.

5.3 Benefits of cloud environment

In this migration, cloud environment was seen to bring positive change to different things. These things were related to the business side of the application as well as technical benefits. Found benefits were somewhat different when asked from different people. Business side benefits were mostly mentioned by the product owner and expert. Technical benefits were mentioned by developers.

Having an application that works in cloud environment is easier to sell for new customers. When the application works on-premises, new customers need to own or buy suitable server machines and SQL Server licenses. In big customer companies, it can be a difficult process to buy expensive server machines. It is likely that the procurement decision needs approval from multiple authorization layers in customer organization. However, cloud application does not need those big financial investments, which can make buying of the application substantially easier.

When the application is sold as SaaS, it is a lot easier to monitor resources than in on-premises environment. As mentioned earlier, this can be done via Azure Portal. Also, it was mentioned as a benefit that scaling instances is easy when the application is sold as SaaS.

If the application is sold to cloud environment owned by the customer, it is still a lot easier to operate than the on-premises counterpart. When the application is installed on-premises, there is often a need to open a remote access connection to the premises of the customer. If the application is in cloud environment, it is possible to access it with just a web-browser.

5.4 Specialties of this migration

When considering cloud migration in this thesis, it means migration where an application is migrated from on-premises to cloud environment. This process contains only phases for converting the application to work in cloud environment. Business requirements are thought to stay similar, no matter which environment the application is working on.

From this migration definition, it is easy to see the biggest specialty of case migration. This migration was not only about making Aveso Data Quality Tool work on a new platform. This migration also contained merging two applications to one completely new application. If these applications would not have been merged, a lot of work towards the application could have been skipped. For example, the front-end of the application was rewritten from Angular to React. This step would not have been necessary, if it would have been more traditional cloud migration.

Another thing that was quite special in this migration was the motivation. In this thesis, the migration is thought to be the migration of a legacy application. That application is migrated to cloud because it is legacy and should be updated to modern time. However, in our case study, the application was only a couple of years old. It can hardly be said to be a legacy application if it has been created within the last five years. The motivation for this migration came from the need to merge these two applications into a new application.

6 Case results

In this chapter literature-based cloud migration is compared with migration that Aveso did when migrating their Data Quality Tool. First, similarities and differences between migration processes are compared. Then architectural changes in case migration are identified and compared with what was found from literature.

6.1 Process comparison

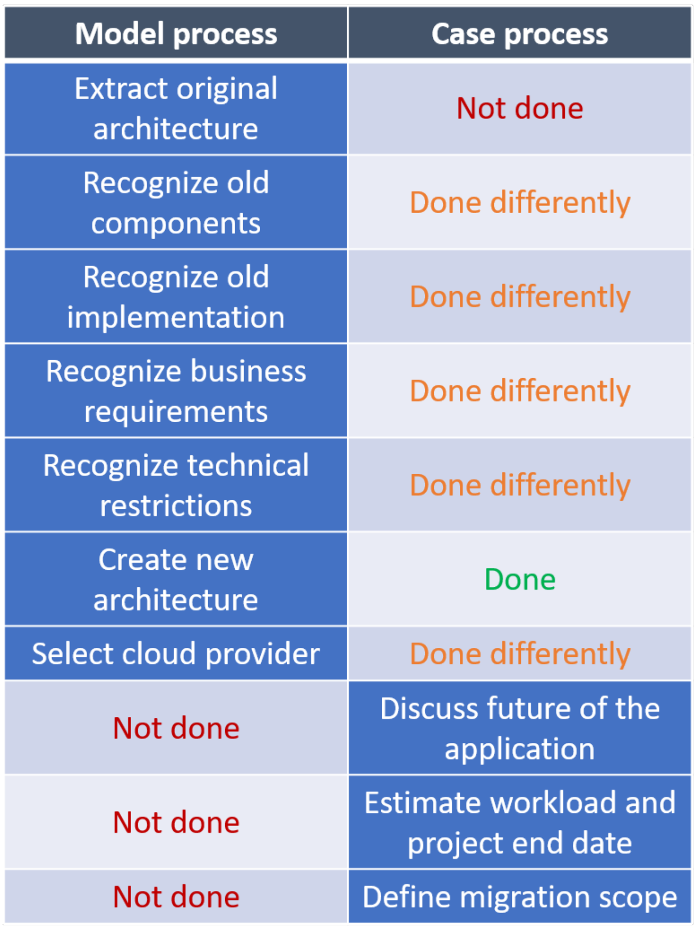
In this thesis, two different migration processes were described. In this section, these two migration processes will be referred to as model process and case process.

1. **Model process** was based on migration models presented in [19], [27]–[29] and it described general migration process. Selected migration models were chosen separately from case process and none of them were used in case process.

2. **Case process** was based on case migration and it described what was done during the case.

Both processes have same phases which are: planning, migration, deployment, and maintenance. Each of these phases will be handled separately. At the beginning of each phase comparison is a figure that compares differences and similarities between model process and case process. From these figures, it is possible to see all the tasks and their origin. For example, in Figure 6.1, task *Extract original architecture* is from model process and it is not done in case process.

6.1.1 Planning comparison



Model process	Case process
Extract original architecture	Not done
Recognize old components	Done differently
Recognize old implementation	Done differently
Recognize business requirements	Done differently
Recognize technical restrictions	Done differently
Create new architecture	Done
Select cloud provider	Done differently
Not done	Discuss future of the application
Not done	Estimate workload and project end date
Not done	Define migration scope

Figure 6.1: Task comparison in planning phase

As shown in Figure 6.1 The only task, that was completely left undone in case process, was extracting original architecture. In model process, the idea was, that the application to migrate is legacy, and the architecture is forgotten or corroded, or both, as mentioned in [28].

Extracting original architecture was not done in case process, because Aveso Data Quality Tool is only a couple of years old application and has not changed that much yet. Also, the application was originally created by the same people who were now migrating it to cloud environment. This means that the architecture was in the minds of project participants.

These reasons make it justified that extracting architecture was not done in this case process. Even though it was not done in this case process, it should be in model process. Extracting architecture is a useful step to have when cloud migration is done for a legacy application.

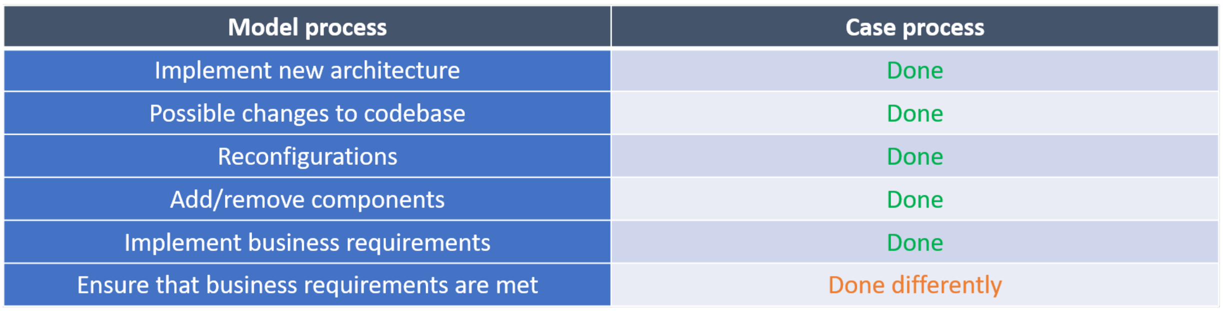
In model process, the idea was that after extracting original architecture, it would be easy to recognize old components, overall implementation, business requirements, and technical restrictions of the application [27]. As said, this task was left undone in case process. Recognizing these things in case process happened through meetings. Meeting participants had a thorough knowledge of Data Quality Tool and also, the documentation of the application was still up to date.

Creating new architecture was something, that was mentioned in model process and also done in case process. Model process mentions, that when designing new architecture, it should be designed in a way that utilizes characteristics of the cloud environment [19], [27]–[29]. Characteristics of the cloud were also considered in the case process.

Final step in model process is selecting cloud provider [19], [29]. This was not done when migrating Data Quality Tool. Aveso had done market research before this migration and found that Azure suits their needs. Also, Aveso is Microsoft's partner company which made it even easier to select Azure. Because of these reasons, research on cloud providers was not done.

Case process had three tasks that were not mentioned in model process. These three are *discussing future of the application, estimating workload and project end date*, and *defining migration scope*. These three tasks are related to each other. Future of the application is discussed to see what should be done for the application overall. Migration scope is then decided. Migration scope defines what will be done during this migration and what can be done afterward. Estimating workload and project end date is dependent on the migration scope. All of these are important tasks that should be in the model process too. Especially, if there are ongoing discussions with potential customers.

6.1.2 Migration comparison



Model process	Case process
Implement new architecture	Done
Possible changes to codebase	Done
Reconfigurations	Done
Add/remove components	Done
Implement business requirements	Done
Ensure that business requirements are met	Done differently

Figure 6.2: Task comparison in migration phase

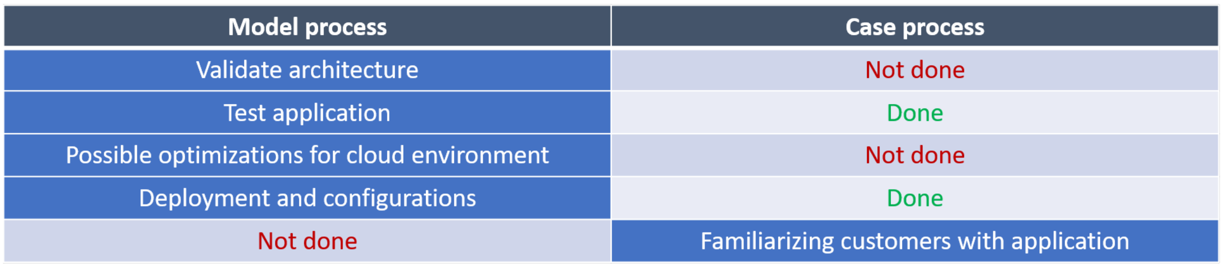
From Figure 6.2 it can be seen that all tasks in model process were also done in case process. In model process it is mentioned that planned new architecture should be implemented in migration phase [19], [27]–[29]. In case process creation of new architecture contained things like adding new components, refactoring codebase, updating used technologies, and changing existing components to cloud equivalents. While doing these things to implement the architecture, it was obviously needed to do some reconfigurations too.

Last two things that are mentioned in model process, are implementing business requirements and ensuring that they are met [27]. In model process the idea is that business requirements are listed before doing the migration. This list can then be used in migration phase to validate that business requirements are met.

In case process, business requirements were not listed at all. Case process had the idea, that the application should be migrated as-is. Therefore, business requirements would be visible in the old system. Also, in the project there were two participants that had good knowledge of these requirements.

Between these processes, there was a difference in how listing business requirements and validating them should be done. Usually listing things explicitly is better than trusting that someone remembers everything. Listing business requirements would have been a better practice in case process too, even though the old system was supposed to migrate as-is. It is possible that the old system contains some hidden requirements, or the requirements are not understood correctly. With explicit listing, these problems could be avoided.

6.1.3 Deployment comparison



Model process	Case process
Validate architecture	Not done
Test application	Done
Possible optimizations for cloud environment	Not done
Deployment and configurations	Done
Not done	Familiarizing customers with application

Figure 6.3: Task comparison in deployment phase

From Figure 6.3 it is possible to see that in this phase there was some differences between processes. Validation of target architecture was the first thing that was mentioned in model process [19], [27]–[29]. However, it was not done in case process. New architecture was created by one person and it was not validated. The architecture was created by following the needs the application had. Therefore, it was thought the architecture does not need to be validated.

Validating target architecture is probably a phase that is often forgotten. Especially, when doing cloud migration without guidance that specifically mentions this. Validating target architecture is still a phase that should be done. It can be that some flaws have found their way into architecture, or maybe the new architecture offers some opportunities that were not thought of when creating the architecture.

Application testing appears in both processes. In case process application testing started with creating test environment. After this, application was tested by using a thorough list of all functionalities in the application. The old version of Aveso Data Quality Tool has a similar list, with the same functionalities, which makes it easy to compare old and new applications. Comparing old and new applications is mentioned in [27] as an important thing when doing tests.

In model process, it is mentioned that when testing an application in cloud environment there might emerge optimization needs [29]. Optimizations are therefore optional and case-by-case. Case process had no need for optimizations, so they were not done. However, it is good to notice that DataPlatform is a new application, and it has not been sold to anyone yet. Aveso will get more data and opinions from daily usage after the first customer starts to use DataPlatform. It might be that some optimization needs are found at that time.

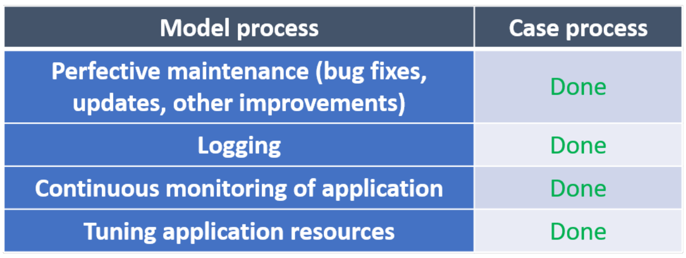
Actual deployment of the application was, of course, mentioned in both processes. In case process the deployment contained few configurations that had to be done. For example, it was needed to connect the application to the data of the customer.

Familiarizing customers with the functionalities of the application was not done in model process. In case process this task contains personal assisting and a user manual. Familiarizing customers with new applications is something that should be in model process too. Though it should be only an optional task, because migrations and their scopes can differ. When doing *rehost* [1] migration, the application will stay just as it was, and it is not necessary to familiarize customers again. However, there are migration cases where the application goes through bigger changes that might affect the functionalities or user interface, or both. For example, *refactor* [2] migration means big changes in the application. These cases are the ones that could benefit from familiarizing customers with application.

[1]Also known as “lift-and-shift”. Introduced in figure 3.1.

[2]Also introduced in figure 3.1.

6.1.4 Maintenance comparison



Model process	Case process
Perfective maintenance (bug fixes, updates, other improvements)	Done
Logging	Done
Continuous monitoring of application	Done
Tuning application resources	Done

Figure 6.4: Task comparison in maintenance phase

Figure 6.4 shows that maintenance phase was similar between processes. In model process it is mentioned that in traditional software applications there is maintenance that can be called perfective maintenance [30]. Perfective maintenance appears in case process as bug fixes, updates to technology, and usage improvements.

Model process refers to a study from R. Marty [31], which states that logs are one of the most important pieces of analytical data in cloud applications. Logs are mentioned in case process too. It is said that logs are used for troubleshooting when a customer has reported a bug.

Continuous monitoring of application services and resources is mentioned in [32] to be one important maintenance task in cloud applications. In this same source, it is said that tuning application resources is also an important maintenance task. Monitoring and tuning are done in case process too. With the case process, it was decided that alerts should be configured to Azure Portal. Alerts can tell you about the state of the application. Besides alerts, case process mentions monitoring by hand regularly. From this monitoring, it is possible to see, if resources need to be tuned or if there is something else that requires action.

6.1.5 Summary

When comparing model process and case process, a lot of similarities were found. It is good to notice that the generalized model process was made based on literature, and migration in case process was done before starting this thesis. Therefore, tasks found in these two processes can be said to be independent of each other.

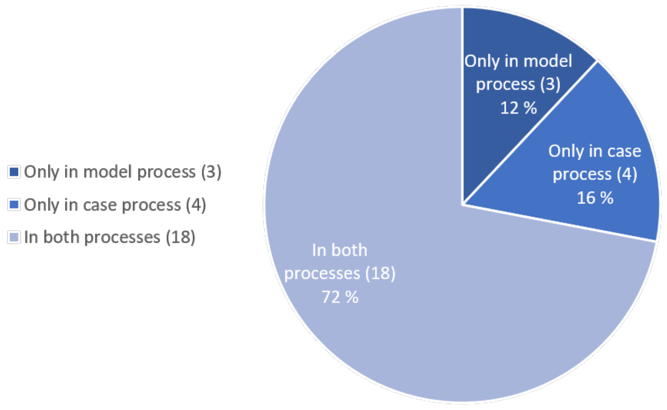


Figure 6.5: Task distribution between model process and case process.

Figure 6.5 shows that there are total of 25 tasks identified in these processes. There were 18 tasks that were mentioned in both processes. Besides these 18 tasks, model process had 3, and case process had 4 tasks that were only in those processes. Tasks that were only in model process were extracting original architecture, validating new architecture, and optimizations to deployed application. It was found that the first two, of those three, should have been done in case process too. The third task was found to be case-by-case, thus the absence of that task in case process was not that prominent, and it still remained relevant for the model process. Tasks that were only in case process were discussing future of the application, estimating workload and project end date, defining migration scope, and familiarizing customers with new application. It was found that model process could have benefited from all four tasks. Though, familiarizing customers with the new application was found to be case-by-case.

After comparing these two processes it was found that model process could benefit from some tasks that were in case process. Figure 6.6 describes model process after adding new tasks from case process to it. Added tasks are highlighted. All tasks in original model process were found to be useful tasks, thus none of them were removed from the model process. Three of them were not used in case process, but they were still found to be useful, at least in some cases that have different starting point than Aveso Data Quality Tool. Case process demonstrated four tasks that are useful in cloud migration, thus they were added to the original model process.

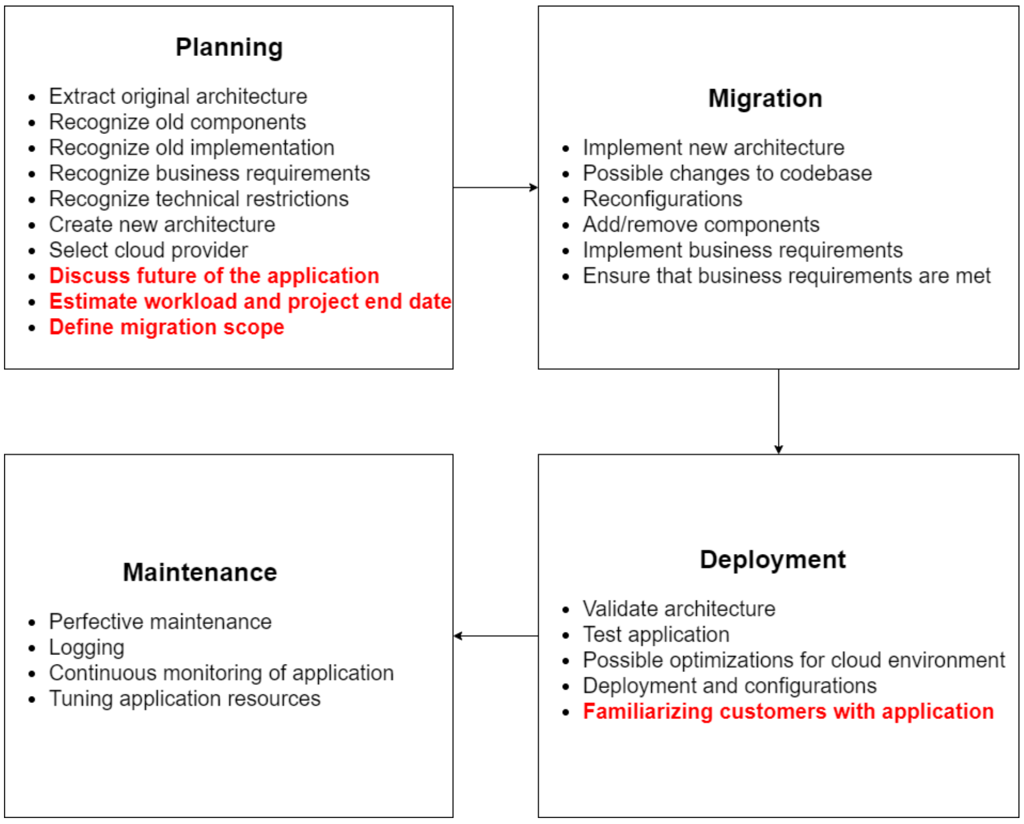

Planning
Extract original architecture
Recognize old components
Recognize old implementation
Recognize business requirements
Recognize technical restrictions
Create new architecture
Select cloud provider
Discuss future of the application
Estimate workload and project end date
Define migration scope
Migration
Implement new architecture
Possible changes to codebase
Reconfigurations
Add/remove components
Implement business requirements
Ensure that business requirements are met
Deployment
Validate architecture
Test application
Possible optimizations for cloud environment
Deployment and configurations
Familiarizing customers with application
Maintenance
Perfective maintenance
Logging
Continuous monitoring of application
Tuning application resources

Figure 6.6: Model process after adding tasks that were found from case process.

6.2 Architecture comparison

During this migration, Aveso Data Quality Tool was merged with Aveso DataHub and migrated to cloud environment. This process caused a lot of changes to the architecture of Aveso Data Quality Tool, as seen in section 5.2. However, most of the changes to the architecture were caused by restrictions from cloud environment.

As mentioned in section 5.1.1, windows authentication could not be used in cloud environment. Because of this, a new component was added to handle user authentication. Another restriction that cloud environment placed, was about executing data quality rules. Old version of Aveso Data Quality Tool handled these executions with database functionalities that are not available in cloud counterpart. Because of this, two more components were added that would handle scheduling and running these executions.

In section 3.2.1 six migration strategies are presented. These are rehost, replatform, repurchasing, refactor, retain, and retire. Replatforming is described in [22]–[24] as a migration strategy where one or more application layers are converted to utilize PaaS. Replatforming strategy allows changes to the architecture, so cloud offerings can be utilized more efficiently.

During the migration of Aveso Data Quality Tool, three new components were added to the architecture. In addition to that, all application layers (presentation layer – business layer – data layer) were migrated to use PaaS offerings. The application now uses App Service plan and WebJobs from Azure instead of the IIS software server running on-premises. In data layer, the application uses a database from Azure SQL Server instead of a database from an on-premises SQL Server. Considering these changes, it can be said that the migration of Aveso Data Quality Tool was done with the replatform strategy.

6.2.1 Relationship between migration strategy and process

Besides architecture, the selected migration strategy also affects whether the model process is needed at all. Based on migration strategy descriptions in section 3.2.1, it can be said that retain, retire, repurchase, and rehost are migration strategies that have low effect on the architecture whereas replatform and refactor strategies have higher effect on it. Figure 6.7 describes how much migration strategy effects architecture. Uppermost have the least effect and lowermost have the most effect.

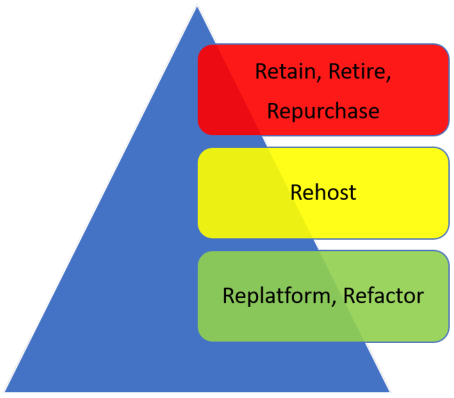


Figure 6.7: Migration strategy effect on architecture and suitability of model process.

1. In *retain* strategy, the migration is not done now. Therefore, model process is not needed at all.

2. In *retire* strategy, it is found that the application is no longer in use. Therefore, the application is retired and migration is not done.

3. In *repurchase* strategy, the organization needs to only find a replacement application for the old application. It can be useful to go through the planning phase, to see requirements for the new application, but migration, deployment, and maintenance phases are not relevant.

4. In *rehost* strategy the application is migrated as-is. Profitability of model process in these migrations is case-by-case.

5. *Replatform and refactor* migration strategies require a lot of work and present changes to the architecture of the application. These strategies will benefit the most from model process. This was seen with case migration too. It was done with replatform strategy and the process was really similar to model process.

7 Conclusions

The object of this thesis was to view different migration strategies and to create a generalized description of migration process from the literature. These objects can provide important knowledge due to increasing interest in cloud migrations. Based on this object, three research questions were presented. Research question (RQ) 1: What tasks have to be done in cloud migration process? RQ2: What different types of cloud migrations there are and how they affect the architecture of the application? RQ3: Which cloud migration types benefit from model process?

Different migration strategies were discussed in 3.2 to answer RQ2. It is found that in literature cloud migration strategies are often presented with six R's. These six R's are Rehost, Replatform, Repurchase, Refactor, Retain, and Retire. They describe shortly what types of migrations there are, and their possible impacts on the architecture of the application. Generalized migration process was introduced in 3.3 to answer RQ1. This migration process contains four phases that are planning, migration, deployment, and maintenance. These phases were built according to what was found in the literature. Generalized process used sources from literature that were chosen separately from case process and none of them were used in case process.

Generalized migration process was compared with the migration, that Aveso did when migrating their application Data Quality Tool. Generalized migration process could be depicted with 21 different tasks. It was found that 18 out of 21 tasks were present in case process. Case process also introduced four tasks that were not included in generalized process. It was found that these tasks are useful and could be added to generalized process too. To give a more thorough answer to RQ1, Figure 6.6 shows generalized migration process, with added four tasks.

The high resemblance between generalized process and case process suggest that generalized process was outlined well. Also, it can be said that literature offers good support for cloud migrations since it was possible to create an accurate generalized migration process based on literature. It was noticed that in some of the migration models in the literature, the learning curve was steep. This was because these models required the user to understand multiple different models to be able to use the actual migration model, as mentioned in section 3.3. While they provide an accurate description of how some phases can be done, it can be exhausting for the model user. Thus, the main takeaway from this thesis is generalized migration process, which has an easy learning curve and is validated with case migration. Price for this easier migration process is decrease in migration phase description accuracy.

To answer RQ3, section 6.2.1 discusses the relationship between migration strategy and migration process. It is found that cloud migration types, that require a lot of work and changes to the architecture of the application, benefit the most from generalized process. For example, case process was identified to be done with the replatform strategy, and it would have benefited from generalized process. The high resemblance between generalized process and case process also support this proposition.

Probably the biggest restriction in this thesis was how the generalized process was validated. This was done with a case that was already almost finished. It would have been interesting to apply the generalized process to some case process from the start. Case migration was done with no help from academic research. Therefore, it would be interesting to do another migration with help from academic literature and then compare these two migrations. Then it would be interesting to see if an academically defined migration process would bring added value to cloud migration. If it would bring added value, it would be interesting to see how much and what type of value.

References

[1] A. Agarwal, S. Siddharth, and P. Bansal, “Evolution of cloud computing and related security concerns”, in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, 2016, pp. 1–9.

[2] T. G. Peter Mell, “The nist definition of cloud computing”, 2011, pp. 2–3.

[3] *Cloud service models (image only)*. [Online]. Available: `https://www.uniprint.net/en/7-types-cloud-computing-structures/` (visited on 10/25/2020).

[4] *What is azure*. [Online]. Available: `https://azure.microsoft.com/en-us/overview/what-is-azure/` (visited on 10/17/2020).

[5] *Azure products*. [Online]. Available: `https://docs.microsoft.com/en-us/azure/?product=featured` (visited on 10/17/2020).

[6] *Azure portal*. [Online]. Available: `https://docs.microsoft.com/en-us/azure/azure-portal/azure-portal-overview` (visited on 10/17/2020).

[7] *Azure app service*. [Online]. Available: `https://docs.microsoft.com/en-us/azure/app-service/overview` (visited on 10/17/2020).

[8] *Azure webjobs*. [Online]. Available: `https://docs.microsoft.com/en-us/azure/app-service/webjobs-create` (visited on 10/17/2020).

[9] *Azure sql.* [Online]. Available: `https://docs.microsoft.com/en-us/azure/azure-sql/azure-sql-iaas-vs-paas-what-is-overview` (visited on 10/17/2020).

[10] *How to connect to azure sql.* [Online]. Available: `https://docs.microsoft.com/en-us/azure/app-service/tutorial-dotnetcore-sqldb-app?pivots=platform-linux` (visited on 10/17/2020).

[11] P. Buxmann, T. Hess, and S. Lehmann, "Software as a service", *Wirtschaftsinformatik*, vol. 50, no. 6, pp. 500–503, 2008.

[12] S. Medhat, "Client/server computing-an engine for change and growth", in *International Seminar on Client/Server Computing. Key Note Addresses*, vol. 2, 1995, 1/1–120 vol.2. DOI: `10.1049/ic:19951146`.

[13] J. Saetent, N. Vejkanchana, and S. Chittayasothorn, "A thin client application development using ocl and conceptual schema", in *2011 International Conference for Internet Technology and Secured Transactions*, 2011, pp. 260–265.

[14] S. Bibi, D. Katsaros, and P. Bozanis, "Business application acquisition: On-premise or saas-based solutions?", *IEEE Software*, vol. 29, no. 3, pp. 86–93, 2012. DOI: `10.1109/MS.2011.119`.

[15] P. Rabetski and G. Schneider, *Migration of an on-premise application to the cloud: Experience report.*

[16] I. Pelle, J. Czentye, J. Dóka, and B. Sonkoly, "Towards latency sensitive cloud native applications: A performance study on aws", in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 2019, pp. 272–280. DOI: `10.1109/CLOUD.2019.00054`.

[17] I. Al-Azzoni, L. Zhang, and D. G. Down, "Performance evaluation for software migration", in *Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '11, Karlsruhe, Germany: Association for Com-

puting Machinery, 2011, pp. 323–328, ISBN: 9781450305198. DOI: 10.1145/1958746.1958792. [Online]. Available: https://doi-org.ezproxy.utu.fi/10.1145/1958746.1958792.

[18] H. A. Müller, “Reverse engineering strategies for software migration (tutorial)”, in *Proceedings of the 19th International Conference on Software Engineering*, Boston, Massachusetts, USA: Association for Computing Machinery, 1997, pp. 659–660, ISBN: 0897919149.

[19] P. Jamshidi, A. Ahmad, and C. Pahl, “Cloud migration research: A systematic review”, *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 142–157, 2013.

[20] T. Boillat and C. Legner, “Why do companies migrate towards cloud enterprise systems? a post-implementation perspective”, in *2014 IEEE 16th Conference on Business Informatics*, vol. 1, 2014, pp. 102–109.

[21] S. Marston, Z. Li, S. Bandyopadhyay, and A. Ghalsasi, “Cloud computing - the business perspective”, in *2011 44th Hawaii International Conference on System Sciences*, 2011, pp. 1–11. DOI: 10.1109/HICSS.2011.102.

[22] K. M. Kumar, S. K. D., R. P. Sardesai, M. B. S. S. Akhil, and N. Kumar, “Application migration architecture for cross clouds analysis on the strategies methods and frameworks”, in *2017 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2017, pp. 107–112.

[23] N. Ahmad, Q. N. Naveed, and N. Hoda, “Strategy and procedures for migration to the cloud computing”, in *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 2018, pp. 1–5.

[24] D. S. Linthicum, “Cloud-native applications and cloud migration: The good, the bad, and the points between”, *IEEE Cloud Computing*, vol. 4, no. 5, pp. 12–14, 2017.

[25] *The best cloud migration path: Lift and shift, replatform or refactor?* [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2018/03/23/the-best-cloud-migration-path-lift-and-shift-replatform-or-refactor/?sh=790ae2294f51 (visited on 11/20/2020).

[26] *6 strategies for migrating applications to the cloud.* [Online]. Available: https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/ (visited on 11/20/2020).

[27] P. Mohagheghi, A.-J. Berre, A. Henry, F. Barbier, and A. Sadovykh, "Remics- reuse and migration of legacy applications to interoperable cloud services", vol. 6481, Jan. 2010, pp. 195–196. DOI: 10.1007/978-3-642-17694-4_20.

[28] S. Frey and W. Hasselbring, "The cloudmig approach: Model-based migration of software systems to cloud-optimized applications", *International Journal on Advances in Software*, vol. 4, Jan. 2011.

[29] P. Pamami, A. Jain, and N. Sharma, "Cloud migration metamodel : A framework for legacy to cloud migration", in *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 2019, pp. 43–50.

[30] R. Mookerjee, "Maintaining enterprise software applications", *Commun. ACM*, vol. 48, no. 11, pp. 75–79, Nov. 2005, ISSN: 0001-0782.

[31] R. Marty, "Cloud application logging for forensics", in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11, TaiChung, Taiwan: Association for Computing Machinery, 2011, pp. 178–184.

[32] V. Podolskiy, H. M. Gerndt, and S. Benedict, "Qos-based cloud application management: Approach and architecture", in *Proceedings of the 4th Workshop on Cross-Cloud Infrastructures and Platforms*, Association for Computing Machinery, 2017.

[33] *Windows authentication*. [Online]. Available: `https://docs.microsoft.com/en-us/iis/configuration/system.webserver/security/authentication/windowsauthentication/` (visited on 11/13/2020).