Tampere University

Tuomo Talvitie

ESTIMATING THE MIGRATION COST TO MODERN CLOUD

An exploratory case study

Faculty of Information Technology and Communication Sciences
Master of Science thesis
February 2020

ABSTRACT

Tuomo Talvitie: Estimating the migration cost to modern cloud
Master of Science thesis
Tampere University
Master's Degree Programme in Information Technology
February 2020

There are multiple reasons for migrating information systems to the cloud. Among the primary reasons are cost reductions, improved scalability and on-demand pricing. The major cloud providers offer various deployment models, and there are also numerous models for migration. Many decisions have to be made during a cloud migration. From this follows that a successful migration requires both knowledge and planning.

The cost of the migration process when migrating systems to the cloud is less well understood than the cost of running the system before and after. This thesis explores the total cost from considering the migration to the finished state. The cost is presented as the time the work takes. The existing literature provides the background information and guidelines for making educated decisions on the migration targets. Estimating the cost of the preliminary work and planning is exploratory, based on a single case study of a software company and previous experiences the author has gained.

A tool for estimating the migration costs was developed along with the case study. The tool guides in the total cost estimation and concretizes and brings visibility to the preliminary costs, as part of the whole migration process. The exploratory nature of the work provides further research targets for increasing the accuracy and guidance the tool could provide.

Keywords: cloud migration, migration cost, cloud services

TIIVISTELMÄ

Tuomo Talvitie: Järjestelmän moderniin pilveen siirtämisen kustannukset
Diplomityö
Tampereen yliopisto
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Helmikuu 2020

Tietojärjestelmiä siirretään pilveen useista syistä. Ensisijaisiin syihin kuuluvat kustannusten alennukset, parempi skaalautuvuus ja kysynnän mukainen hinnoittelu. Suurilla pilvipalvelujen tarjoajilla on useita käyttöönottomalleja, ja pilvimigraatioiden toteuttamiseen on myös useita malleja. Ratkaisuja joudutaan tekemään lukuisia. Tästä seuraa, että onnistuneen migraation taustalla on tietoa ja suunnittelua.

Järjestelmän pilveen siirtämisen kustannuksista on vähemmän tietoa olemassa kuin järjestelmän ylläpitokustannuksista ennen ja jälkeen. Tässä diplomityössä tutkitaan pilvimigraation kokonaiskustannuksia migraation harkitsemisesta migraation valmistumiseen. Kustannukset ilmaistaan käytettyinä työpäivinä. Kirjallisuus muodostaa työlle teoriapohjan ja suuntaviivat päätösten tekemiseen siirrettävistä kohteista. Kustannusten arviointi migraation esityöstä on kartoittavaa tutkimusta, ja perustuu ohjelmistoyrityksen tapaustutkimukseen ja kirjoittajan aikaisempiin kokemuksiin.

Tapaustutkimuksen rinnalla kehitettiin työkalu siirtokustannusten arvioimiseksi. Työkalu ohjaa kokonaiskustannusten arvioinnissa ja konkretisoi ja tuo näkyvyyttä migraation esikustannuksiin osana koko prosessia. Diplomityön kartoittava luonne tarjoaa jatkotutkimuskohteita tarkkuuden ja työkalun tarjoaman ohjeistuksen lisäämiseksi.

Avainsanat: pilvimigaatio, pilvimigraation kustannukset, pilvipalvelut

PREFACE

Cloud services are impressive. Thank you for everyone who has made writing this thesis possible and has supported writing it. Big thank you for Jari Peltonen and Cometa Solutions Ltd for providing information, support, and the primary case for concretizing the ideas I had for the thesis. Special thank you for my wife Johanna for her outstanding support. Big thank you also to Toivo the toddler for all his essential help.

Returning to studies was interesting; it looks like I do not need to do it again. Thank you, Professor Kari Systä, for supervising the thesis.

Tampere, 6 February 2020

Tuomo Talvitie

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
AWS	Amazon Web Services
CI	Continuous Integration
DB	Database
EC2	Elastic Compute Cloud
FaaS	Function-as-a-Service
GCP	Google Cloud Platform
IaaS	Infrastructure-as-a-Service
IS	Information System
MCET	Migration Cost Estimation Tool
PaaS	Platform-as-a-Service
SaaS	Software-as-a-Service
VM	Virtual Machine

.

1. INTRODUCTION

This Master of Science thesis discusses the various migration costs [acquiring expertise and knowledge, and the time it takes, planning, testing, simultaneous running of environments] encountered when migrating a functioning information system (IS) to the cloud services, or when considering such migration. A notable part of this thesis focuses on the knowledge, skills and expertise required for successful cloud migration, and the work and time required in acquiring them. While multiple papers exist on cloud migrations and strategies pertained, few consider the migration cost and no papers attempt to quantify the cost of acquiring the necessary knowledge and expertise.

While the cloud as definition contains various services, this thesis concentrates on the computing part of cloud computing, in a similar vein as Boza *et al.* in their paper [1], with the inclusion of storage, including databases. The migration targets modern cloud providers with options for usage-based charging and load balancing, and serverless capabilities. The migration may require architectural changes to acquire cloud compatibility or cloud compatibility. Migrating legacy software is an additional cost factor which this thesis also discusses.

There are recognized benefits to being cloud-native, an application designed to take advantage of the properties of the cloud. These advantages and properties of cloud nativity will be discussed later in this work. While it is worth considering if the end goal is for the system to be cloud-native, the effort required may be too high to be cost-effective. The following chapter goes through some of the main migration types and related cloud services.

The basis of this thesis is existing literature and research, with a single case study approach for exploratory research into the preliminary costs of cloud migration. The case focuses on aspects that affect the total cost of migration. The Cometa Solutions case is fruitful for this thesis based on the years of evolution the system has gone through, making it a useful example of a nontrivial existing system. A brief discussion of two additional cases motivating the creation of this thesis follows the primary case. They support the exploratory research, introducing some experience report aspects to this single case study.

The work, along with the primary case, includes the creation of a tool for estimating migration costs on a high level. The purpose of the tool is to provide a template for estimating the total cost of successful cloud migration. As a novelty, the tool includes acquiring cloud readiness, defined as the knowledge and skills required by the migration, and the cost of acquiring it. The areas covered are the preliminary work, planning, implementation, and maintenance and evolution.

The thesis has been divided into chapters as follows. Theoretical background in chapter 2 follows this introduction. This background information describes the aspects of the cloud and cloud migration necessary for understanding the rest of the thesis. Chapter 3 covers the cost estimation process. This process leads to chapter 4, documenting the Cometa Solutions case. The case includes brief background information about the software architecture, the solutions built on top of it, and the migration cost estimates. The two supplementary cases follow the primary case in chapter 5. The estimation tool and cases are then analyzed in chapter 6. The thesis finishes with the Conclusions in chapter 7.

2. SOFTWARE SYSTEMS AND MIGRATING THEM TO CLOUD SERVICES

This chapter provides background information on cloud services and migration options relevant to understanding the rest of the thesis. The basis is the basic understanding of cloud services and cloud computing, benefits and pricing. Internalizing specific cloud concepts, such as cloud-native, docker and microservices, is essential to understanding how the migration options described in the R's of cloud migration affect the system.

A significant subset of papers on cloud migration describes migrating legacy systems. These systems cause problems to the organizations that depend on them. Migrating them can also carry additional costs. These legacy issues are discussed briefly in the subchapter 2.6.

2.1 Cloud services and cloud computing

The cloud by itself is a rather broad term, and in the context of this thesis, is used to refer to cloud computing, including storage and databases (DBs). The definition by NIST for cloud computing from 2011 is ubiquitous with thousands of referrals. Briefly, the essential characteristics are

1. provisioning of resources as on-demand self-service
2. availability over network
3. pooling and dynamical assignment of resources
4. elastic provisioning and releasing to match demand
5. resource usage monitoring and measuring.

Together with the characteristics above, the model consists of three service models and four deployment models described below. [2]

2.1.1 Service models

There are three service models, with distinct abstraction levels, offered in the cloud: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and SaaS. Starting from IaaS, where cloud provider manages the actual servers and storage, networking and security, and the actual data centre, each of the services expands the services provided in the cloud, with SaaS being complete application running on a cloud infrastructure. The service models and the aspects managed by the cloud provider can be seen in Figure 1 below. [2]

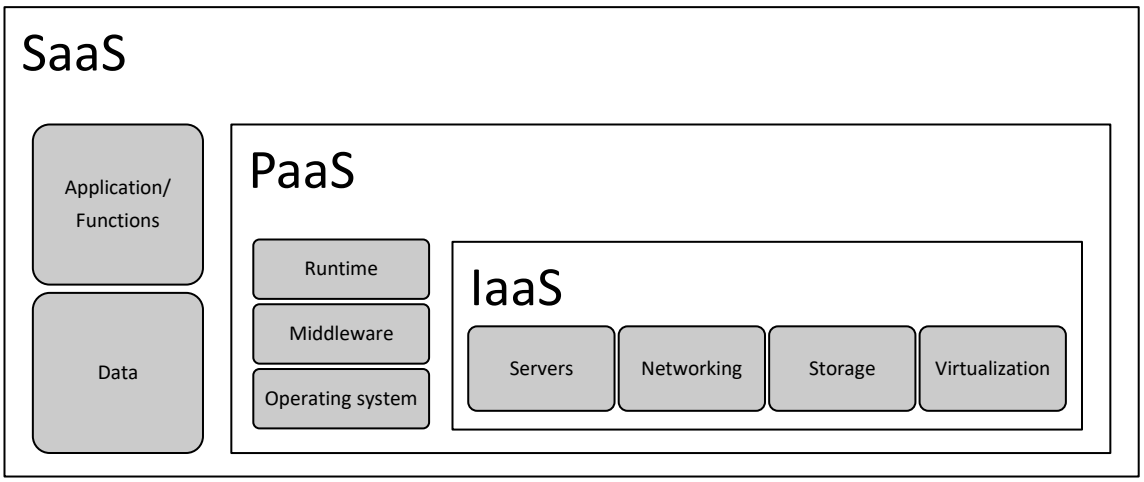

Figure 1. *Cloud models – SaaS, PaaS, IaaS*

IaaS – Infrastructure as a Service

IaaS is the low abstraction level of cloud services, where clients do not control the actual infrastructure, but can provision computing resources and control the operating system, deployed applications and storage [2].

Examples of IaaS are Amazon EC2, Google Compute Engine, Windows Azure

PaaS – Platform as a Service

PaaS builds on top of IaaS, abstracting away the operating system and infrastructure, but may control the configuration for the environment hosting the application. The platform provides everything for creating, hosting and deploying an application executable in the environment supported by the provider. [2]

Examples of PaaS are AWS Elastic Beanstalk, AWS S3, Google App Engine, Heroku, Azure Cloud Services.

SaaS – Software as a Service

SaaS allows running applications provided by their provider, and abstracts away even individual capabilities of applications, allowing only limited user-specific configuring of the application [2].

Examples of SaaS are G Suite. Microsoft Office 365 and Salesforce.com.

2.1.2 Deployment models

The NIST definition lists four deployment models for the cloud infrastructure. Private cloud, community cloud, public cloud and hybrid cloud. The names describe the provisioning: private cloud for a single organization, community cloud for consumers with

shared concerns, and public cloud is available for anyone. The hybrid cloud is a combination of two or more of the first three, bound together to enable data and application portability. [2] This thesis concentrates on migration to the public cloud. Nevertheless, outside the procuring and management of the cloud, the thesis can be adapted to any deployment model.

2.1.3 Serverless computing

Serverless, also Function-as-a-Service (FaaS), is named after the concept of abstracting the server management away from the clients. The cloud provider manages the execution environments in the background, and the clients pay only for the resources that are used by the functions executed via client requests. In practical terms, the clients pay for the time (in seconds, measured in sub-second precision) a specific amount of CPU and memory is used, with consumed network resources added to the bill. [1] In comparison to PaaS, serverless can scale to zero instances [3].

While SaaS provides complete applications for the end-user, from the perspective of designing an application, and purchasing resources from a cloud provider, it is not a logical abstraction step after IaaS and PaaS. It can be, and it has been, argued that serverless computing is the next abstraction level in the cloud after PaaS [4]. Baldini *et al.* place serverless between PaaS and SaaS, and conclude that the boundaries of serverless are an open research problem [3]. Interestingly, Jonas *et al.* in the 2019 Berkeley View predict that serverless computing will become the default computing paradigm of the cloud [5].

Examples of serverless implementations are Google Cloud Functions, AWS Lambda and Azure Functions.

2.2 Potential benefits

One important reason for the popularity of cloud solutions is money saved on fixed costs due to leasing rather than buying infrastructure [6]. Another reason is that SaaS, PaaS and IaaS are universally accessible, acquirable and releasable on-demand, and payable based on usage – fulfilling the need for high computing capability, scalability and resource consumption [7]. The costs are kept low by the economy of scale and having large-scale data centres at low-cost locations, together with statistical multiplexing to increase utilization [8]. 2.3 Pricing describes the pricing model in more detail.

Further on, the availability of practically infinite computing resources eliminates the need for planning provisioning ahead, and the lack of up-front commitment allows increasing

commitment only when the need for them actualizes [8]. Khanye *et al.* note that cloud migrations affect companies by shifting the focus from operating the environment to being more innovative, as IT personnel gain more versatility and develop business analyst and management skills. [9]

2.3 Pricing

The pricing of cloud services is complicated, with varying pricing models and discounts [10]. A single cloud provider can also have different prices in different regions, such as *us-central1* or *europe-west-6*. The GCP Cloud Spanner is an illustrative example of this with hourly prices of $0.90 and $1.17 for the above examples [11]. On the positive side, the competition is driving the cloud compute prices down, historically about 25% a year. [10]

The pricing of the three major cloud providers is on-demand based, with calculators existing for estimating the costs. On the IaaS side, the properties selected for the virtual machines, such as CPU and amount of memory, determine the price. This is then billed by the second, commonly with a minimum charge of one minute. [10] Similarly, the monthly network bill is calculated from the network usage. Data is charged by the stored amount, with a variety of options for access and retrieval speed.

For example, Google Cloud charges $0.12 per outgoing GB when monthly usage is between 0 and 1 TB when using the datacentre (region) in Finland. Incoming traffic is not charged, nor outgoing traffic to the same zone [12]. To highlight this as a variable, the price is $0.09 per outgoing GB when using Amazon AWS Stockholm, disregarding the first gigabyte, which is free [13]. The cost in Azure is comparable with a price of $0.087 per outgoing GB after the first free 5 gigabytes in the North Europe region [14].

As briefly noted in 2.1.3 Serverless computing above, the billing of serverless functions is time-based. The providers have premium services available, but the base price calculation happens by multiplying the cost of selected resources with execution time and the call count. AWS provides an example of allocating 512MB of memory, calling the function with execution time of 1 second 3 million times within a month, which results in a cost of $18.34 [15]. Applying the same settings to calculators of Google Cloud and Azure, the costs end up as $25.15 and $18.00 respectively. The results do not include potential network costs. Scaling up to 30 million calls, Azure costs $239.40, AWS costs $249.18, and Google Cloud costs $285.70. As can be seen, the prices vary but are in the same ballpark in this case.

Considering the pricing examples above, the 38% higher price of the most expensive option (GCP) compared to the cheapest one (Azure) indicates that there are at least temporarily situations where the selection of cloud provider matters economically. The fact that GCP ended up as the most expensive option in both example prices should not be interpreted as a constant, as the prices change and Google has been the initiator of several price reductions [10].

2.4 Cloud concepts

There are some instructive concepts for understanding cloud computing, cloud deployments, migration to the cloud, and the alternatives to migration. On one side is targeting cloud nativity, on the other side is retiring the system. Understanding the cloud concepts will help in making educated decisions considering the work needed, costs, and positive and negative aspects.

2.4.1 Cloud-native applications

Cloud-native is used to describe system built in the cloud with a set of properties that many cloud-native applications have in common. The five most common properties of a cloud-native application, as listed by Gannon *et al.* are [16]

1. operating on a global scale – data and services can be replicated in a robust way in datacenters near end-users to minimize latencies.
2. scaling well with thousands of users – together with the global operation sets requirements on synchronization and consistency.
3. the assumption that failure is constant and the infrastructure is not static – on a global scale, the law of large numbers guarantees that something is broken or about to break, but the application should be able to keep working.
4. built for continuous operation, avoiding disruptions from updates and testing – this requirement sets demands on the architecture.
5. built with security built-in, not an afterthought – the application is often built from small components which must not contain sensitive credentials. Access control management must happen on multiple levels.

Linthicum lists four benefits for cloud-nativity [17]:

6. Performance – use of native features available
7. Efficiency – cloud-native features and application programming interfaces available for improved performance and/or reduced costs
8. Cost – efficiency translates to lower costs due to usage-based pricing
9. Scalability – direct access to autoscaling and load-balancing features

Microservices (see 2.4.3) is the most common approach for building cloud-native applications [16]. However, Kratzke and Quint state that service-based approaches are vital

for cloud-native approaches, and the *micro* is not the essential part. Microservices nevertheless appear to be seen as crucial enablers for cloud-native applications [18]. They end up with the following definition for cloud-native application:

> "A cloud-native application (CNA) is a distributed, elastic and horizontal scalable system composed of (micro)services which isolates state in a minimum of stateful components. The application and each self-contained deployment unit of that application is designed according to cloud-focused design patterns and operated on a self-service elastic platform. [18]"

Multi-tenancy

Multi-tenancy describes the practice of supporting simultaneous requests from several clients running on shared hardware and software infrastructure. Usually, this is achieved using either multiple instances or native multi-tenancy. Multiple instances describe the case where each tenant is a separate instance over shared resources and native describes a single application shared with numerous clients. As can be expected, native multi-tenancy supports more tenants. Multi-tenancy is mostly a matter of cost, limiting the amount of money spent per client, and the problems to be solved are especially cases where there are varied requirements from the clients. Guo *et al.* describe in their paper how the multi-tenancy capabilities could be enabled. [19]

Elasticity

Elasticity is an advanced version of scalability, where the resources are increased or decreased dynamically depending on the current or expected demand. Elasticity can be seen primarily as a cloud computing concept, as it provides the infrastructure of procuring and releasing resources on demand.

Cloud-enabled applications

Gholami *et al.* call cloud migrated system cloud-enabled [7], taking in the definition by Chauhan and Babar defining the migration as software re-engineering that allows the application to interact or integrate with cloud services [20]. Analyzing the work of Chauhan and Babar a bit deeper rises the notion that the critical requirements for a specific cloud-enabled system should be defined based on the system, not necessarily by the available features in the cloud. For example, they identified elasticity as a property of SaaS cloud platforms in their analysis. It became a key requirement for cloud-enabling their application because of the performance requirements of a multitude of projects with dozens of developers around the world. [21]

Jamshidi *et al.* write that software migration is a particular case of adaptive maintenance modifying the system to fit the new environment, and part of the process is utilizing the new features and confirming that the applications keep working [22]. The documentation

of Google Cloud repeats the same thought for successful cloud migration: one should analyse both the migration to cloud and modernization [23].

2.4.2 Docker and Kubernetes

Kratzke and Quint note that the need for standardizing packages of CNA components repeats in several studies [18]. Docker, a de facto standard fulfilling this need, allows automated deployments of applications in self-contained deployment units [18], a kind of lightweight virtual machines, running on a host system. Kubernetes is a cluster manager for Docker containers, created by Google [24]. Kubernetes has emerged as a de facto tool in the space of container management, load balancing, and storage orchestration. Since David Bernstein's article from 2014, when Amazon did not yet have complete Kubernetes support, all major cloud providers now support Kubernetes. [25]

2.4.3 Microservices

Architecture based on microservices separates the system into small services that can be deployed and scaled independently [26]. This is the opposite of a monolithic way of building a system. The scalability allows for efficient use of cloud services, as close to optimal resources based on the demand can be purchased from the cloud provider. The downside of microservices is that while single service can be simple, modifying an existing system to microservices is rarely straightforward. Creating one from scratch takes extra work, as the distribution of the business logic is a complex task, requiring several components [26].

Extended discussion on microservices and microservice architectures is outside the scope of this thesis. However, when considering microservices *Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation* by Taibi *et al.* can be recommended as a source of information. Some of the key findings are briefly explained below. The article also includes three processes practitioners use for migrating monolithic systems to microservices [27].

Benefits

Based on the empirical investigation by Taibi *et al.,* the improved maintenance, in the long run, is the most important benefit. Migration consultants stressed improved scalability, but this was not as important to others. These two benefits acted both as drivers for migration and were also reported as benefits afterwards. [27]

Disadvantages

The primary issues encountered by practitioners when migrating to microservices-based architectural style are in order: The complexity of decoupling a monolithic system, migration and splitting of data existing in databases, and communication among the services. Additionally, DevOps infrastructure, found to be necessary by all participants in the study of Taibe *et al.,* requires effort on top of the development effort. [27]

Taibi *et al.* report that the initial costs are higher for a microservices-based system than a more traditional one, which they note matches the findings by Singleton and Killalea. This initial extra effort was, however, reportedly compensated after one to three years due to reduced maintenance costs. [27]

2.4.4 Cloud provider learning resources

The three largest cloud providers are AWS, Azure and Google Cloud. All the major cloud providers offer resources for learning and experimenting with their offerings. The details vary, but in practice, a newly registered user has 12 months of limited free tier usage, credits, or both to use instead of actual money during that period. The description of AWS, Google Cloud and Azure are listed below.

AWS Free Tier

“The AWS Free Tier provides customers the ability to explore and try out AWS services free of charge up to specified limits for each service. The Free Tier is comprised of three different types of offerings, a 12-month Free Tier, an Always Free offer, and short-term trials. Services with a 12-month Free Tier allow customers to use the product for free up to specified limits for one year from the date the account was created. [28]“

Google Cloud Platform Free Tier

“The Google Cloud Platform Free Tier gives you free resources to learn about Google Cloud Platform (GCP) services by trying them on your own. Whether you're completely new to the platform and need to learn the basics, or you're an established customer and want to experiment with new solutions, the GCP Free Tier has you covered.

The GCP Free Tier has two parts:

- A 12-month free trial with $300 credit to use with any GCP services.
- Always Free, which provides limited access to many common GCP resources, free of charge. [29]”

Azure Free Account

“The Azure free account includes free access to our most popular Azure products for 12 months, $200 credit to spend for the first 30 days of sign up, and access to more than 25 products that are always free. [14]”

Use of free tiers

As stated in the descriptions above, the free tiers allow for experimenting, learning and testing new solutions. They lower the threshold for cloud migrations, as the cloud services can be safely experimented with, by for example, creating the infrastructure on low-end machines, and testing network configurations required for the migration. As only used resources are paid for, there is no need to make significant monetary commitments to the platform until one is ready to do so.

Learning resources

An interesting take on learning the basics, even some advanced features, is provided by educational services, such as Qwiklabs [30], which offer automatic courses with access to the relevant Cloud resources included in the packages. In the case of Qwiklabs, there exist resources for learning the use of both Google Cloud and Amazon AWS cloud resources. The Google Cloud Free Tier currently includes some credits for the service, allowing taking courses for example from the basic understanding of provisioning virtual machines (reserved time for the lab 40 minutes [30]) to learning Kubernetes in Google Cloud (5 hours, total) [31].

2.4.5 Other considerations

The migration time is, in the author's opinion, an excellent time to review deployment strategies and tooling, including continuous integration (CI). This is however outside the scope of this thesis. The CI basics can be studied at for example CircleCI’s documentation. [32]

2.5 Cloud migrations

The benefits of the cloud and cloud migrations have been discussed above in 2.2. However, moving the existing architecture to the cloud as-is will not, in most cases, bring those benefits. [26] This thesis emphasises the how and why of the migration process. However, one should not forget doing a comprehensive analysis of the costs and benefits. Linthicum states that figuring out the correct path for cloud migration is the most challenging part [17]. For example, moving a system from server hardware to a virtual machine will seldom bring scalability improvements. However, if the environment is not

in constant use, the server might be shut down, for example, weekends, potentially bringing costs down (see Pricing 2.3 above).

2.5.1 Determining what to migrate

A variation of a 3-tiered design is typically the basis of an existing enterprise application [33]. The decomposition of the application includes a front end tier, a business logic tier and a back end tier for datastore. The back end tier can be, for example, a relational database. Thus, there probably exists at least two components that can be considered for migration separately, removing the need to migrate the whole system. In *Cloudward Bound*, Hajjat *et al.* have counted that an enterprise application has between 11 and 100+ components [33]. When considering targets, it is possible that, for example, testing environments exist which may be a more straightforward starting point than the production systems.

When selecting the components for migration, one must be aware that the components may depend on each other, and transactional delays must be considered if components, or their users, exist in different locations [33]. These transaction delays before and after the migration, among other concerns, are considered in the model by Hajjat et al., who stress the importance of planning when making migration decisions [33].

2.5.2 The R's of cloud migration

The various amounts of "R's" describe the options available for a system under consideration for cloud migration. Gartner outlined the R's, Rehost, Refactor, Revise, Rebuild and Replace in 2011 [34]. They have been revisited since then. During 2016 Orban wrote about six different migration strategies, 6 R's, seen implemented by customers [35], and Linthicum wrote about 7 R's in 2017 [17]. The strategies are described briefly below, along with a note on which R lists they appear. The list below includes all the items, with some overlap and some conflicting descriptions. However, as the distinct items are mostly sound, in the end, the list contains 9 R's, with perhaps some potential for consolidation.

Rehosting – also known as lift-and-shift (5R, 6R, 7R)

The rehosting strategy minimizes the changes to the application and only replicates the current system on the cloud platform, hence the lift-and-shift moniker. Nevertheless, according to Orban, GE Oil & Gas saved approximately 30 per cent by rehosting [35]. He also states a finding that the optimization/re-architecting is more manageable after the

migration is complete due to experience and having the application, data and traffic already in the cloud [35].

Replatforming – "lift-tinker-and-shift" (6R, 7R)

This strategy optimizes the system for the cloud (or otherwise), for example by moving databases to managed services or integrates cloud monitoring as part of the system. The core architecture stays as is. [35] This option is included in the 5R as part of refactoring option [34].

Repurchase/replace (5R, 6R, 7R)

This option most commonly moves the desired functionality to a SaaS-based application, for example, from CRM to Salesforce.com [35]. Existing data typically requires migrating [34].

Rebuild (5R)

This strategy involves building a new cloud-native application while discarding the existing source code [17,34]. 7R includes this as part of the replace option.

Refactoring / re-architecturing (5R, 6R, 7R)

In this case, architectural changes are made, usually with cloud-native features. Refactoring is potentially the most expensive option, but also one that can have the most benefits. Business needs for features, scale or performance not achievable by the current system are the typical reasons for selecting this option. [35]

Revise (5R)

This option from the 5R is a combination of several other options, with first modernizing the codebase and then refactoring or rehosting [34]. The author of this thesis considers this option somewhat mismatching compared to the other options, and notes that this option does not exist in the subsequent version of the "R's".

Retire – remove (6R, 7R)

Not everything stays useful, and potential savings exist in removing unneeded applications [35].

Retain – do nothing (6R, 7R)

Cloud migration should not be executed if it does not make sense for the business. This option also includes revisiting the decision later. [35]

Reuse (7R)

Reuse is the option for either consolidating similar applications and services or breaking an application apart for reusable components and developing shared business and technical services. [17]

-

The different options have different amounts of effort required as well as opportunities for optimizing the solutions (agility, scalability, …). Linthicum argues that the benefits of cloud-native applications outweigh the cost, and even argues that not making an effort is a mistake [17]. A general idea on the relative amounts of work in each of the R's can be read, for example, from the AWS migration whitepaper [36].

2.5.3 Migration types

In their paper, Zhao and Zhou compare the various migration strategies identified in other sources and end up with three main strategies. The strategies are on a high level the ones described above, IaaS, PaaS, and SaaS with three sub-strategies: replacing by SaaS, revising based on SaaS, and reengineering to SaaS [37]. The first sub-strategy is known as Repurchase from the R's discussed in 2.5.2. The second sub-strategy involves partially replacing some functionality with cloud services, a close fit with replatforming R, and the third is again one of the R's, re-factoring / re-architecting the system to the cloud. The strategies involve various amounts of work; it is essential to consider the scope of the migration, and the cost-benefit analysis on the possibilities.

Migration to IaaS

IaaS migration is conceptually simple, Vu and Asal state that the cost can even be free if the requirements are consistently stable and match the existing service plans. The statement, however, presumes extensive knowledge on the selected cloud platform, including network routes and hardening, as they also clarify. These they call hidden costs and also include installation and even administration costs. [38] The challenge of migration to IaaS is the detection of resource usage and management [38]. A simplistic but potentially effective way to do this is to note the potential targets that may not need to be running 24/7, for example, some testing or build environments. Shutting those down while not needed, allows taking benefit of the time-based billing, as only the used storage resources are billed. (See 2.3 Pricing above.)

Migration to PaaS

Vu and Asai state that PaaS migration does not contain the hidden costs of IaaS due to the working on a higher level. The resource management is automated and can be based on current demands from the applications. [38]

There are both general guides for PaaS migration and guidelines restricted to the specific PaaS providers. These include checklists for specific requirements and solutions for problems, such as solving incompatibilities with database migrations [37].

Zhao and Zhou state that PaaS offers a complete cloud IT stack for building cloud-native applications for scalable and elastic environment. However, restrictions exist at every technology layer, of which Zhao and Zhou list five below [37]:

1. Programming languages. This restriction is also picked by Vu and Asai as the most critical checking step, as the support for languages is limited [38].
2. Databases. This restriction affects migration only if there are dependencies to specific database [38].
3. Middlewares
4. Third-party libraries
5. Restrictions specific to the selected PaaS

Migration to Serverless

A complete serverless migration necessitates using a serverless architecture, requiring changes to the existing logic. The statelessness of serverless functions requires initialization of any database connections and outside data sources on each instance, increasing response times. For example, in Ruutiainen's MSc thesis, the application logic was rewritten to remove statefulness, which is not always possible. [39] Ruutiainen states that the running costs of serverless applications are minimal, but the costs are incurred from the development [39]. A conclusion was presented that the serverless is best used as a part of a larger system, instead of forcing the whole system to fit the paradigm [39].

Migration to SaaS

In case the migration can be executed with the replacement strategy from the "Rs" using commercial SaaS service, there is no need for reengineering, when replacing the whole system, and the migration effort is significantly reduced [37]. Zhao and Zhou list this as the first sub-strategy for SaaS migration. The second sub-strategy replaces partial functionality of the system with an existing cloud service. The third identified sub-strategy re-engineers the system to cloud service. [37]

The re-engineering strategy, refactoring/re-architecturing from the "Rs", can be very challenging. Zhao & Zhou list potential work to include reverse engineering, structure redesign, service generation, and more. [37]

2.5.4 Requirements for migration, planning of migration

Before planning the migration, its feasibility should be ensured. Vu *et al.* have analysed the specific constraints and limitations that prevent migration to cloud and have separated and targeted these concerns for PaaS and IaaS migrations separately. [38]

The requirements for IaaS are less restricting than for PaaS because, in IaaS, users can run their preferred operating system and software on virtual machines (VMs). In PaaS, limitations exist in the form of programming languages and databases supported by the platform [38]. The situation is however improving, and for example, Google's App Engine currently supports Java, PHP, Node.js, Python, C#, .Net, Ruby, and Go, having added .NET in 2017 [40]. There can also be processing time restrictions on PaaS [38].

2.5.5 Migration strategies

On a high level, the cloud providers have a vested interest in successful migrations to the cloud. To facilitate this, they have created some instructive white papers that condense the questions and possible solutions into easy to follow packages. For example, *the CIO's Guide to Application Migration* by Google contains within 20 pages the whole migration process. The overviews and diagrams guide the selection of an appropriate migration strategy and target [41]. When viewed from a more generalized perspective, the answers are readily generalizable to any cloud environment. In general, the author of this thesis recommends reading the above whitepaper to get general information, and then reading *Strategies and Methods for Cloud Migration* by Zhao and Zhou to get a more granular (and non-commercial) view of migration strategies and methodologies [37].

Alternatively, to Google's offering, the *Cloud Migration Strategies* [42] for Azure contains the instructions needed for migration in simple steps, with the technologies picked from Azure offerings. However, here as well the steps themselves are generally applicable – starting from assessing the environment and continuing to evaluating how best to migrate each application.

2.5.6 Vendor lock-in

The potential cost caused by vendor lock-in needs to be considered as well. Opara-Martins *et al.* state that the complexity and cost of switching often come apparent at the implementation stage [43]. The research by Opara-Martins *et al.* highlights the lack of awareness of proprietary standards prohibiting interoperability and portability [43].

Despite the concerns, it does look like that competition is driving prices down and improving feature parity, even when the offerings are not directly transferable. The basic features on IaaS, PaaS, even on serverless are of comparable level. The differences may be more evident on specialized services such as machine learning and natural language offerings.

2.5.7 Risks of cloud migration

Gholami *et al.* collected from literature five reasons migration have sometimes failed to achieve the goals set for it [44]:

1. Lack of understanding of requirements of cloud computing
2. Technical implementation started too early
3. Lack of planning
4. Seduction by hype
5. Unexpected issues out of control of both service consumers and providers

Linthicum covers similar issues in his presentation [45] and recommends against starting cloud migration with mission-critical or legacy systems. The lack of knowledge and experience is also one of the main problems, along with not testing and verifying technology choices.

2.6 Additional considerations with legacy systems

Literature defines a legacy system as “any information system that significantly resists modification and evolution” [46]. This definition from 1995 is quoted, for example, by Bisbal *et al.* [47] and Khanye *et al.* [9]. The definition is rather old but fitting. Three out of the four main problems identified at the time are still valid. The cost of maintaining obsolete hardware is perhaps not the most pressing issue in today's servers, outside exceptional cases. One could argue that today this should be replaced by the lack of scalability, a cost in today’s environments, as explained further on. However, the cost of maintaining the software and difficulties in tracing faults due to the lack of documentation and lack of understanding on how the internal systems work is a valid problem, and a factor encouraging the cloud migration. Another of the problems is the absence of clean interfaces. Moreover, crucially, it is difficult, perhaps impossible, to expand the legacy systems. [48] The negative aspects of legacy affect the cost of migration by either adding work or by making it harder to take advantage of the advantages of the cloud, such as elasticity.

The problems mentioned above translate to costs in cloud systems. The problems that affect cloud migration costs are explained below. The problems are, in many ways, the flip side of the benefits associated with cloud-native (see 2.4.1).

Lack of scalability and elasticity

A legacy system may make it difficult to react to changes in increased or decreased system resources, for example, an increased number of users, or new resource-heavy processes. A legacy system may be able to scale up with more powerful hardware, but scaling out with node increase may not be supported [44]. Older systems may have been created before techniques such as load balancing have been commonly available [7]. If elasticity is needed, the system must be refactored to support it [44].

Lack of documentation and understanding of the system

Lack of understanding increases the cost of migration, as legacy business knowledge and design architecture need to be recovered. [49]

3. COST ESTIMATION

Taking in the areas discussed in the chapter above, a guideline for estimating the cost of cloud migration can be developed. As stated before, the cumulative time taken by the preliminary work, migration planning and execution is considered the main factor here. All the major cloud providers have cost calculators, where calculating the running costs is somewhat trivial once one knows what to input in the forms. The running cost may be prohibitive in cases where the resource or network usage is severely high and required resources currently already exist, but that is a business decision outside the scope of this work.

The total cost can only be accurately known after the migration is complete. However, as the cloud migration is no longer a novelty, plenty of research into it has already been done by academia and the cloud providers, as can be seen above in this work. Based on these works, and the author's own experiences with companies planning and executing cloud migrations, estimation on the necessary tasks for a successful cloud migration can be created. Further on, the minimum effort of those tasks can be estimated, with a guide for improving the estimate based on the specifics of the migrated system. The accuracy of the estimations certainly vary depending on the cases they are applied against. However, by defining the result for the task, the estimate can be improved in any specific migration case.

The cost estimation results can be used to balance against the expected benefits, and educated decisions can be made on whether to continue with the migration. This decision can be made at several points during the estimation.

3.1 Challenges in estimating the migration cost

The main challenges identified during the research are the comparable lack of literature of costs encountered during the migration, as also identified by Antohi [50], and the lack of literature including the cost of acquiring the skills and knowledge for the migration at all. Due to this, the estimates on acquiring the skills and knowledge is mostly explorative, based on experience built with the cases.

Further on, a fixed approach to migration is not applicable for all migration scenarios, but based on literature review by Gholami *et al.* [7] there exists little work instructive in designing custom approaches that match the characteristics of a migration project. Thus,

by necessity, estimating the migration cost requires at least a rough planning of the actual migration.

3.2 Sources of migration costs

The migration costs can be divided into three areas. The cost of acquiring knowledge and experience, planning and executing the migration (migration work), and the costs after the migration.

The cost of the migration work comes from a mixture of sources. Here, as can be expected, the labour costs account for a large portion of the migration. The reasoning is simple: the cloud migration process is work; from analysing the current solution, to planning and executing the migration, potentially requiring new implementations of solutions.

In addition to the migration work costs, running the cloud environment(s) have a cost attached to it. The cost should be estimated and compared to the original system at the planning stage. The running costs can be lower after migration. However, confirming the feasibility, it should be affirmed that both the cost of the migration and the upkeep afterwards are within an acceptable range to avoid surprises. Antohi states in *Model for Cloud Migration Cost* that the cloud vendors offer pricing sites capable of simulating the cost of the system architecture in the cloud [50]. However, the migration costs are outside their scope. Khanye *et al.* condense the result succinctly: “However, the findings suggest that cloud savings is not primarily on financial costs as one still pays, but for different things.” [9]

3.2.1 Acquiring necessary knowledge and skills for migration

As discussed earlier, there are quite many steps in successful cloud migration. Most require some specific knowledge or skills to complete. Understanding of the current architecture is crucial in finding out the best result and adapting the system to cloud. Moreover, the level of adaptation may directly affect the cost of running the system in the cloud.

Selecting the cloud provider(s) for the migration is part of the migration process. Gholami *et al.* identified multiple factors affecting cloud provider selection [7]. These include the service models offered, price model, monitoring, auto-scaling, supported programming languages, and much more. The major cloud providers and learning resources can be found in 2.4.4 above, which may help in gaining an understanding of the services provided.

Having experience and information on cloud practices is advantageous in the migration process and contribute to the cloud readiness of the system and organisation. Here, cloud readiness describes the total readiness for the migration, including the knowledge on what to do during the migration as well as an understanding of the current system and architecture, so that the migration can be expected to succeed.

To ascertain the readiness, the cloud adoption readiness test created by AWS may be helpful. One should note the disclaimer stating that the organization is responsible for making an independent assessment. The questionnaire is useful also as a checklist; for example, a question on the test asks whether a strong understanding of operating securely in the cloud exists. [51]

3.2.2 Planning the migration

During the planning exists a decision point on what exactly are the objectives for the migration. The objectives limit the possible migration types available. The list of “Rs” discussed in section 2.5.2 covers the primary alternatives on how to proceed. Merely moving the software to IaaS is the simplest solution, using the rehost strategy. However, most of the benefits of the modern cloud in regards to elasticity may not be available. Cloud compatibility and the work involved is discussed in more detail in 3.2.3 below.

The restrictions, existence of guides and checklists helpful in guiding through the process were briefly discussed in 2.5.3–2.5.5 above. Regulations, privacy laws and latency requirements may also restrict the migration [44]. Risks (see 2.5.7) can prevent it. Thus, the planning should include the possibility of rolling back at any point of the migration to reduce risks [44].

Understanding the migrating application is crucial. Acquiring the necessary knowledge may require mapping dependencies, architecture and functionalities. The quality of documentation can significantly affect the time and cost of this investigation. [44]

The cloud migration reference model (Cloud-RMM) by Jamshidi *et al.* from a compact systematic literature review includes the following tasks under the planning [22]:

1. feasibility study
2. requirements analysis
3. provider and services selection
4. migration strategies

A significantly more detailed view can be read from the systematic review by Gholami *et al.* [7] and is recommended for a complete insight into cloud migrations. For an informative overview, the Cloud-RMM can be recommended.

3.2.3 Modifying the system as cloud-compatible

One of the significant aspects affecting the cost of cloud migrations is the cloud compatibility of the applications. The existence of various restrictions was mentioned in 2.5.4 above. A cloud-compatibility analysis may identify incompatibilities with the cloud computing environment(s) being considered for migration. For example, the target cloud may lack support for frameworks used by the migrating application, and this must be solved. [20] The work that resolves the found incompatibilities can include code refactoring, developing integrators or adaptors, and data adaptation [44]. The properties of cloud-native applications, some or all which may be applicable here, were discussed in section 2.4.1 Cloud-native application.

The cost of modifying the systems cloud-compatible is the cost of planning, including identifying and analysing the requirements, combined with the actual work of modifying the systems and then deploying them to the cloud. Vu and Asai note that in case of migration to PaaS, the cost of migration is the cost of adopting the application to run on the PaaS [38], which is true, but may include much work.

3.2.4 Data storage and transfer

Information stored in the existing system may be invaluable to the organization. In other cases, the system may have some specific requirements or dependencies on the data storage. In some cases, the amount of data may be substantial. In all cases, the current system and requirements need to be verified and mapped to the cloud offerings. Depending on the level of documentation on the current system and knowledge on hand, this may take some time. For example, a single Drupal front-end system may contain language definitions in a version-control system, data in document storage in various formats, and data in a database. In case the system, and data, is migrated, all these must be considered.

General data storage in the cloud

There are various ways to store data in the cloud, with different levels of abstractions and methods of access. The basic types of cloud storage are block storage, object storage and file storage. Block storage is comparable to traditional hard drives, object storage refers to the storage of objects, uniquely identified and accessible for example through https, without hierarchy where the file system is abstracted away. File storage is a file system that is managed by the cloud provider with the storage device abstracted

away. The advantage of these in comparison to traditional servers is the immediate availability of the storage and the fact that even with block storage, modern cloud providers bill only based on stored data.

Part of the understanding of the migrated application (see 3.2.2) is finding out the data usage, including the size and operations per second. This knowledge helps in understanding the workload being migrated to the cloud and in estimating the cost of cloud usage. [7]

Databases in the cloud

Databases, in a similar vein as the general data storage, come in managed and less managed varieties, relational, non-relational, and in-memory varieties. The number of options allows for selecting an optimal service, but on the other hand, requires balancing the requirements, the cost of the service and the cost of adapting the current system to the service. The DB migration tools provided by the cloud vendors, such as *Azure Database Migration Service* [52] or migration guides such as *Migration from MySQL to Cloud SQL* [53] may help in the migration.

Data transfer

The cost of data transfer can be an essential consideration, as the cloud providers bill by the amount of data transferred. Traditional servers may have fixed cost, whether in-house or hosted externally. The transfer costs are primarily a concern for the application running in the cloud, as incoming traffic is generally free, and thus the transfer costs during the migration are not an issue. However, the amount of transferred data and how long the transfer takes can be an issue.

3.3 Estimation process

When considering the total cost of cloud migration, the time that planning and knowledge gathering takes cannot be ignored. Moreover, there is always the first migration, where the cloud will at least partially be unknown waters for the organization. It is even possible that no cloud services have been used, and only the concept of the cloud is known. This state is the starting point for the cost estimation in this thesis.

It is possible to split the complete process into various different stages. In this thesis, there are four distinct stages identified in the complete process:

1. preliminary work
2. planning
3. implementation

4. maintenance and evolution

Preliminary work contains tasks considered to be necessary for analysing the system and ensuring that required knowledge and expertise exists for the migration to succeed. Completing the steps creates the knowledge basis for the migration. The completion also allows making an educated decision on cancelling the migration if the case for it is not strong enough. The planning stage contains further analysis and estimations, with the result of a road map, which guides the migration through the implementation stage. Finally, the system should be maintained and developed further. The complete list of tasks and the Migration Cost Estimation Tool (MCET) in tabularized form can be found in Figure 2 in section 3.4. The creation of the tool is explained in the same section as well.

3.3.1 Starting from the basics

Understanding the cloud is the basis for making appropriate decisions on the migration process. Thus, the starting point is learning the basics of the cloud, using the teaching material included in the Free Tiers of the cloud providers. Any of the major cloud providers are suitable for this, although one should read the introductory material from each of them to familiarize oneself with the differences and offerings, which may contain some specialized cloud services suitable for specialized systems. In case cloud expertise already exists, learning the basics can certainly be skipped.

The second success factor is the knowledge of the system being migrated. The knowledge is required to make educated decisions during the planning stage with the awareness of their results and effects. If the knowledge, expertise or documentation is lacking, they must be acquired.

Together these tasks make up the Preliminary work step in the MCET. The estimated total here is nine days, which can be lower in case of existing knowledge on the system and its requirements and the cloud, and higher in case the system is complex and/or there is little to no existing cloud knowledge.

3.3.2 Planning for success

The process continues in the planning phase by identifying the features of the existing system relevant to the migration. The MCET covers these under the Planning topic. Sources of migration costs 3.2 above covers the essential topics investigated here.

Understanding the system and cloud environments is essential for the planning stage. The knowledge allows making migration decisions component by component, without adversely affecting the system if such a path to cloud is decided, and the application

components are decoupled [7]. The same applies, in lesser effect, if the whole system is moved to the cloud. The migration strategies must nevertheless be decided based on the expectations, requirements and time limits. The yield here is a roadmap for a hopefully successful migration, along with a strengthened understanding of the cloud and the total system - even if the migration is not executed.

Considering the costs of running the system in the cloud, helpful tools in estimating the total cost exists in the form of papers, and related tools, along with the cloud provider calculators. For example, Cloud Migration Point model can give an estimation of the size of a cloud migration project, if the necessary assumptions, such as all design decisions having been made, are fulfilled [54].

The planning step estimation in the MCET, with 4 components identified as targets, is estimated to take at least 15 days. Here again, extensive existing knowledge will reduce the time, and oppositely more work in analyzing the components, or in the knowledge discovery and architecture reviews, will take more time.

3.3.3 Implementing the plan

Following the roadmap will most likely bring some surprises, but having prepared, these should be possible to overcome. Thus the changes in the architecture, code and handling of data should be straightforward work. While the implementation step tries to offer some idea on how long the tasks will take to roughly estimate the total cost of migration before it has been started, more accurate information should be available from the created road map.

In this thesis, the testing of the migrated system is not a separate step from implementation, as experience indicates that they are not readily separable. Here, the matter is about the definition of done, a question discussed in agile development, and it can be argued that the implementation is not over until the migration is Done. See, for example, Derek Huether's short essay on the subject [55]. However, a final validation, once everything is otherwise complete, does make sense, although how this should be implemented and on how rigorously, differs case by case.

The implementation step here is the one that can take much longer than the minimalistic time estimation described here as an example of a simple lift-and-shift migration. The four-component project is expected to be implemented in four days with proper planning and prior experimenting in the cloud. The lift and shift strategy is the lightest of the migration R's where actual migration occurs, and as such, can be used as a measuring stick. It is quite likely that more time is needed in most real-life migrations. How much

time depends on the amount of work needed, and the roadmap should estimate the extensity of code modifications and any other changes to the system.

3.3.4 Maintenance and evolution

The final step here is the maintenance and evolution of the migrated system(s). This is included, as it is unlikely that the migrated system, despite all the planning, will be the optimal solution, especially cost-wise. One of the practices available is to on purpose migrate to compute resources known to be overpowered. Then, once everything is tested and running, analyse the actual resource usage in the cloud and apply that knowledge to reduce costs. This method allows for deferring the optimization to a later stage. On the negative side, this gives a less accurate picture of the running costs at the planning stage, but on the plus side, the likelihood that the actual running costs will be more than the planned ones is reduced. The process of optimizing the running instances is called right-sizing [56].

The maintenance also includes some vital testing related activity: data verification, backup testing and creation of a recovery plan [7]. It can be argued that the data verification could also be part of implementation or the possible separate testing stage discussed in planning.

Modernization is also included as an ongoing task, as the cloud services are being developed, and other related technologies as well. This task is not as such part of the migration, but rather a reminder for trying to avoid the system falling into legacy status.

3.4 Estimation tool – MCET

The Migration Cost Estimation Tool condenses the total information presented in this thesis in a simple tool. The purpose is to present the tasks required by a cloud migration with notes on required information to be able to fulfil the task successfully. The tasks include a rough estimation of the time required for the task. Executing tasks will allow improving the estimations of the following tasks, thus improving the accuracy. The tool is not a complete guide for a cloud migration project, but can, perhaps supported with this thesis and some of the recommended reading, at least help in clarifying the extent of the project.

3.4.1 Tool creation process

The motivation for the thesis originated from two cloud migration projects. Detailed descriptions of these cases are not significant, but the general descriptions and some pertinent details are presented later on in chapter 5 Additional cases. Together these two cases roused the interest in the question of the work required, and cost involved, for successful cloud migration. Due to timing related issues, the use of these two migration projects as the primary cases was not possible.

The process of writing this thesis began with a literature review, which resulted in five primary findings:

1. There exists a lot of literature on cloud migrations.
2. A significant portion involves legacy systems.
3. The models for migration appear to be rather diverse, with the actual models often targeting very specific migration target.
4. Few of the research papers available discuss the cost of the migration work in any detail. This is discussed in 3.1 Challenges in estimating the migration cost.
5. Few of the research papers discuss the required knowledge for a successful migration.

Due to the amount of literature combined with the lack of sources for costs, the selection of sources for this thesis was based on several literature searches with various keywords (cloud migration, costs of cloud migration, legacy cloud migration, …). From the results, items were selected for further review based on

1. Title or abstract indicating suitability (cost of migration, migration work, migration process)
    a. Titles indicating VM migration or only running costs were discarded
2. Number of times cited
3. Publishing date

From these sources, combined with personal experiences and web sources, the tool was created by listing the primary tasks, the knowledge required, and the rough estimates of costs. The contents were improved iteratively during the case study described in the following chapter. The cost estimation process can be qualified as explorative, as necessitated by the lack of existing research.

The guiding principles for the development of the tool were

1. having the necessary cloud information for planning to be effective
2. the need for planning before implementation
3. the iterative improvement of the accuracy of the estimated cost as knowledge increases by the practitioners.

These principles by design avoid risks 1–4 listed in 2.5.7 Risks of cloud migration, and the planning ahead and knowledge gathering attempts to mitigate the unexpected risks.

3.4.2 Tool description

In practice, the cloud migration consists of the steps contained in the table below as described in 3.3 Estimation process above, with estimates of the duration each step takes. The granularity for the duration estimates of the tasks is approximately half a day. The tasks are based on the information sourced from research, as described in chapters 2 and 3, and combined with the cases the author has been involved in. The estimated durations are exploratory, educated, but rough estimations based on experience, and thus subject to change.

The general assumption in the given duration estimations is that the person(s) working on the project have some cloud knowledge, and little to none practical experience, but have in-depth knowledge of some part of the system being migrated. Any additional knowledge can be included as a reduced time estimation in the MCET.

An unverified assumption, based on the supplementary case A, is that more people involved in the project does not decrease the time. Instead, the opposite is expected in cases without prior cloud experience. However, the result may still improve with more brains involved, and more in-depth knowledge gets cross-pollinated. An expert can reduce the amount of time spent. On the other hand, if the current system is close to being a black box for the project participants at the beginning of the migration project, all categories will take additional time.

The tasks included in the estimation tool, in general, can be seen to match the Cloud RMM migration framework by Pooyan *et al.*, although the model expects the requisite knowledge and skills acquired in the preliminary work in this thesis to already exist. [22]

	Topic	Skills / knowledge	ID	Tasks	Duration
Preliminary work	Cloud principles	Cloud provider account, services	Pre1	Sign in to free tier of cloud provider, execute learning tasks	1-3 days
	Migration strategy	Understanding of migration process	Pre2	Read Google Whitepaper, Zhao&Zhoa, and more	1 days
	Potential targets	(Logical) architecture	Pre3	Identify high level targets	1 days
	Requirements	Requirements	Pre4	Identify the requirements of the targets	1 day / target
	Potential benefits	Current costs, downsides	Pre5	Case for migration	1-2 days
	Cloud readiness	In-house experience	Pre6	Identify knowledge, expertise	1 day
	"	"	Pre7	Identify gaps in knowledge and expertise	1 day
	"	review of cloud readiness	Pre8	Go through AWS CART (or similar)	1 day
	"	"	Pre9	Plan and acquire missing information, required expertise	1-x days
Planning	Existing data	Data	Pla1	Identify data and requirements	1 day
	Current architecture, environment	Architecture	Pla2	Identify features of current architecture, environment	1-3 days
	Analyse migration targets	Logical architecture	Pla3	Identify migratable systems and their components	1-2 days
	Migration strategy	Migration strategies, 6 R's	Pla4	Determine migration strategy/strategies	0,5 days / component
	Migration work estimation		Pla5	Analyse work needed for migration	0,5-1 days / component
	Cloud provider features, pricing		Pla6	Cloud provider selection	3 days
	Analyse costs		Pla7	Translate the requirements and cloud architecture to migration costs	2 days
	"		Pla8	Based on the generated knowledge, calculate running costs in cloud	1 days
	Roadmap		Pla9	Create roadmap	2-5 days
Implementation	Implement		Imp1	Follow roadmap	1 day + 0,5 days / component*
	Test and validate		Imp2	Test that the system is working and fulfils the requirements	1+ days
Maintenance	Optimize		Mai1	Rightsizing, ++	1 day+
and evolution	Modernization		Mai2	Consider cloud nativity, containers, best practices	ongoing

* lift-and-shift. Automated tools may help here. Duration increased as a function of refactoring needed.

Figure 2. Migration Cost Estimation Tool

4. CASE COMETA

Analysing the system and the starting point of cloud migration of Cometa Solutions made the creation and gradual improvement of the estimation tool possible. The case is presented as follows: First, the background information useful for understanding the case and its place in this thesis is presented. The background includes the origin of the solution and high-level architectural information of the system. This is followed by information on the solutions and their usage profiles. Information on how these are developed and deployed is then described, and the aspects affecting the migration described. Finally, the cloud migration cost is estimated, and the process of using the estimation tool briefly evaluated.

4.1 Cometa description

Cometa Solutions creates digital environments based on a metamodel based platform. Systems built on top of the platform contain shared features, such as automatic version control, comprehensive access management and real-time propagation of data between users. [57,58] The solutions are, in most cases, provided as SaaS.

4.1.1 History of Cometa's software architecture

The Cometa Solution's software architecture originates from the concept of everything being a model element. Moreover, these model elements are based on a common meta-meta-model (a model of a model of a model). Domain models are specialized from this base model. This affects the architecture, as all information is contained within elements based on archetype elements of the metamodel. While this complicates the base implementation, the platform allows component reuse on all levels, as once implemented features can be used on any system as the base model provides the blueprint for any component, even UI-elements.

One important evolutionary step of the architecture has been discussed in the MSc thesis of Vesa Lahdenperä, in which the web-view architecture was designed and implemented in the model-based environment. The architecture has continued evolving since then, incorporating separate notification services into the mix, but the thesis is still in essence accurate description of the general principles of the solution. [59]

4.1.2 System operating principles

As stated above, the basis of the system lays in the meta-meta-model, from which domain models are specialized. These are used along with specialized models for roles and privileges, and state machines, creating a coherent environment where all components are reusable no matter the actual domain model, as the meta-meta-model is always the common denominator. Attached to this is the UI-model taking advantage of the other models, with tools for creating user interfaces from building blocks. Together this allows for WYSIWYG (What You See Is What You Get) style user interface creation.

The current architecture allows the system to be built upon using the concept of cells, a kind of components built upon respective domain models, which can be specialized from their basic general implementations. The concept enables customized software to be built with a relatively low amount of actual customization work.

4.2 High-level architecture

The current logical architecture can be seen below in Figure 3. The system is divided into several services that communicate with the database. The underlying architecture of the backend running the services can be defined as layered. Additionally, the client running on the end-user devices is based on a layered architecture. The database is replicated onto the slave, which can be accessed as read-only for queries if needed.

One of the consequences of using meta-meta-model, and the domain-specific meta-models, is that by definition and design, it is a shared domain model. This is a concern with microservices where coupling may be problematic; ideally, each microservice can be developed independently from each other. Considering cloud nativity, the database can be distributed outside a single master, but changing the whole architecture to microservice based would add unwanted complexity. Service layer components support scaling out with additional nodes.

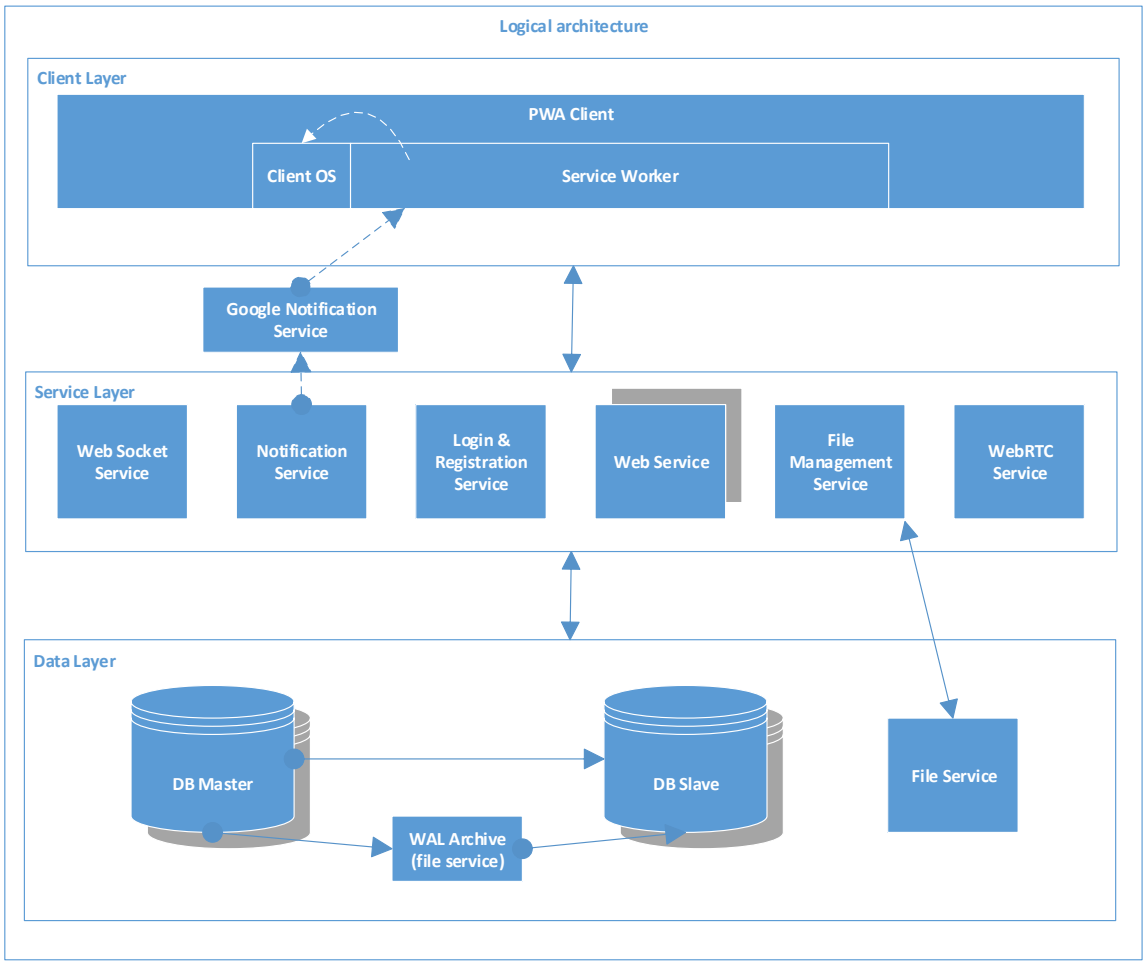

Figure 3. *High-level logical architecture*

4.3 Usage profiles

There are currently three product segments and their corresponding usage profiles served by the production system. The profiles have distinct usage models that define the resource requirements of the system. These profiles are, with a bit of generalization, information management, webinar and analysis solutions. The resource usage and requirement profiles are described below.

4.3.1 Information management solutions

Information management solutions, content creation and editing, generate small transactions to the database and scale up with the number of users. Larger files are saved into a separate file server. Some of the queries for information are complex and thus resource-heavy, such as user information searches from multiple locations.

An existing information management system has a thousand users, and a maximum of one hundred simultaneous users. Here, most functions executed are not slow, but a long-running report exists, which can take 20 seconds to execute. History data for changes are saved in a self-built temporal database. Performance-wise the queries into history are expensive, but a read replica can be used to alleviate the resource usage on the master database. The profile is presented in a table format in Figure 4 below.

The information management solutions make use of Web Socket Service, used for notification about changes in the models, uses noticeable amounts of memory, little CPU or disc space, and utilizes Google Notification Service. Can be a bottleneck, and modification could be applied to limit the number of listeners to models

Category	Source of resource usage	Use of resources	Effect on requirements
Tasks	Content fetches	Small read transactions	Small up to hundreds of simultaneous users
	Content creation and editing	Small write transactions	Small up to hundreds of simultaneous users
	Large file transfers	Network, storage	Separate file/object storage required along with DB
	Information queries	Slow reads	Memory requirements for caching
	Report generation	very slow transaction - rarely	Some memory, CPU required
	Notifications	listens to changes in model´s	Significant memory use
Users - total	1000 users	created content in DB	Little to no effect
Users - simultaneous	100 users	execution of tasks above, mainly fetches	Little to no effect
History data	History enabled	Heavy DB read queries	Must not affect primary tasks -> Read replica / CQRS

Figure 4. *Information management – example profile*

4.3.2 Webinar solutions

Webinar solutions, the profile presented in Figure 5 below, allows transferring of screen, sound and inputs between users. These create a lot of internet traffic that scales up linearly with users. WebRTC [60] is used for transferring desktop and sound, and as multicast is not possible, the traffic scales up linearly with the number of participants. For robustness, the data is relayed through Cometa's server, which makes the network usage potentially a significant source of expenses in the cloud.

Category	Source of resource usage	Use of resources	Effect on requirements
Tasks	Screen sharing	Network	A lot of pass through traffic
	Voice sharing	Network	Pass through traffic
	Video sharing	Network	Pass through traffic
	Notifications	listen to models	Small
Users - total	200 users	created content in DB	Little to no effect
Users - simultaneous	30 users	execution of tasks above	Network traffic calculation below
History data	History enabled	A little to record	Little to no effect

Figure 5. *Webinar solutions – example profile*

Network traffic can be estimated by noting that the codec used is VP8, and using the bandwidth estimation seen in Table 1 below [61].

Table 1. *WebRTC bandwidth estimations per stream*

Mode	Streaming Rate
4k	15–20 Mbps
1080p	4–8 Mbps
720p	1–4 Mbps
VGA (640x480)	500k–1 Mbps

For 8 users in a webinar session, we can calculate the outgoing bitrate by adding up the appropriate streaming rates multiplied by the receivers. Thus, using the maximum values, the upper limit for the resource usage can be calculated:

```
1 presenter to 7 receivers at 1080p
   → 7 x 8 Mbps = 56 Mbps
7 receivers, each sharing their faces at 640x480 to each other
   → 7 x 7 x 1 Mbps = 49 Mbps

Together, the figures above total 105 Mbps.
➔ 1 hour at 105 Mbps equals to 47.25 gigabytes.

With 30 users, extrapolating from the 8 users' figure (the 30 simultaneous
users will not be in the same webinar) the streaming rate equals approxi-
mately 400 Mbps.
➔ Equalling 180 GB in an hour.
```

This is a significant amount of traffic, which must be considered when considering the cloud migration. Note: most cloud providers rate only outgoing traffic. See 2.3 Pricing. For example, the 47.25 GB equals to $5.67 with the Google Cloud pricing for the data-centre in Finland.

4.3.3 Analysis solutions

Analysis solutions involve large transactions, although not continuously, creating a potential DB bottleneck and using significant amounts of CPU resources. The analyses are executed, when possible, outside operative use in replicas.

Category	Source of resource usage	Use of resources	Effect on requirements
Tasks	Expensive queries	Extensive reads	DB memory
	Data mining - data loading, transforming, statistical modeling	Reads, calculations	CPU, DB memory
	Large file transfers	Network, storage	Separate file/object storage required along with DB
	Information queries	Slow reads	Memory requirements for caching
Users - total	-	-	No effect
Users - simultaneous	10 users	Tasks above	Multiplies the work, memory essential to avoid slowing everything down
History data	No writes - history data may be part of analysis	Heavy DB read queries, there may be a lot of history data	DB memory , CPU

Figure 6. *Analysis solutions – example profile*

4.3.4 Common features

For the most part, the users come from Europe, during the daytime. No users are expected from elsewhere, except temporarily. Furthermore, as stated in Cometa's privacy documentation, all data is stored and handled within Europe.

Together these pieces of information on the usage profiles can be used in estimating the need for resources in each solution. Having this information is useful in understanding the logs on resource usage and extrapolating the need for additional resources if the number of users would increase, for example.

4.4 Development and deployment

The development process can be divided into four steps on the implementation side. The first steps are executed on the computer of the developers, where the UI-client is running on the developer's computer. The backend and databases are running on the common development environment. As the feature being developed is being implemented, the development server is updated at suitable points, allowing feedback collection during the process. Once the feature is deemed ready for more comprehensive testing, automatic testing included, the system is built and deployed on the testing system. And once the feature has been validated, the same is done on the production system

The testing system and release system have the same configurations; the only difference is the provided resources. The development system does not have as high availability requirements, which is its main difference from the other systems, along with the reduced requirements for being similar to the production system.

The cloud migration in the Cometa case can be considered an environment at a time, keeping in mind the requirement for the testing environment to match the production environment. Thus there are three logically separate environments to consider for cloud-compatibility. They are the development, testing and release environments, and each has a distinct purpose and requirements. The process of releasing into production is described in Figure 7 below, from development through testing into release.

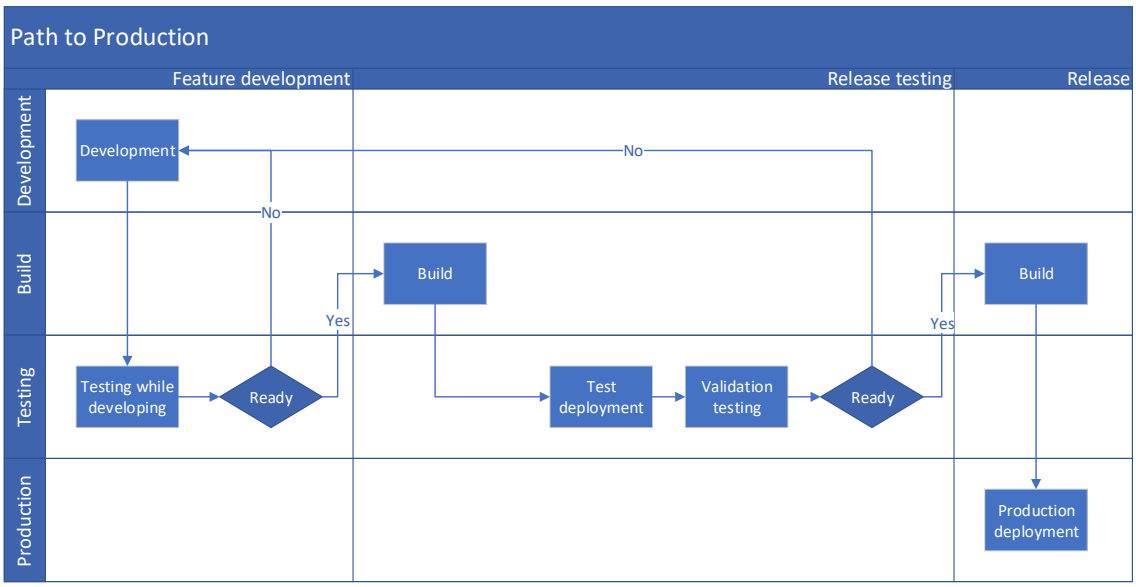

Figure 7. *The path from development to production*

The three environments are architecturally similar. The similarity allows the introduction of cloud benefits and tailoring of the migration process to the requirements to get the most benefits with the least amount of work. In case the system is already close to being cloud compatible, analysis can be done once and applied to all of the systems. As the environments are nevertheless identical and have separate requirements, the quirks of the different systems should be noted, documented, and resolved.

4.4.1 Cloud benefits

The cloud migration analysis was motivated by the possibilities of efficient scaling and provisioning cost-effective environments quickly as needed. The current hardware resources and hosting are very competitive, especially with the network resources currently being a fixed cost, so the primary motivation was not the cost of the resources.

The system was analysed to identify the potential benefits of cloud migration and the costs associated with enabling the benefits. The migration was considered to slightly reduce the time taken by administrative tasks, such as updates and backups, as managed solutions will be kept up to date by the host. The flexibility of the options and inbuilt monitoring of resources was one of the larger plus sides.

4.4.2 Aspects affecting the migration

There are four main aspects affecting potential migration:

1. Human resources
2. Architecture by design not microservices
3. Existing infrastructure both computationally powerful and costs competitive
4. Currently, infrastructure and information can be guaranteed to stay in Finland

Cometa has limited human resources with cloud experience, and these resources have limited availability for the project. Some of the knowledge is also considered to be not entirely up to date. Before more in-depth analysis, the time estimation for moving the production system, as-is, to a cloud provider's system is 1.5 person-months, in calendar time 2 to 3 months. The estimation is with the expectations that the resulting system would be correctly configured and secured in the cloud environment using the best practices approach.

As discussed in section 4.2 High-level architecture, the architecture of the system is based on meta-models, and as such microservices can in practice only be used to supplement the architecture, and distributed systems and scaling up depends on the services and the models the services are part of.

The existing infrastructure is running on reasonably powerful hardware, and the administration costs are also reasonable. In addition to this, both the internal and external networks are fast, and most importantly, have a fixed price. The infrastructure is located in Finland, which is an important factor for some customers.

4.5 Estimating the cloud migration cost

Going through the tasks identified for successful migration gives us an estimate that can be used to make an educated decision on the profitability of the migration. As the estimation tool was developed along with the case, the results documented here contain some work that can be considered extraneous compared to the normal use case. In these cases, this is elaborated on in the description.

Here, each of the tasks in the estimation tool in relation to the Cometa Case is described with either time taken or an estimation of it. The information is presented in a slightly generalized form to be readily readable to practitioners and to allow for easier comparison of experiences in relation to similar cases.

4.5.1 Preliminary work

Pre1 Sign into a free tier of a cloud provider, execute learning tasks

During the work, accounts were created in AWS and Google Cloud and got familiar with the environments. Due to having some experience with AWS beforehand, and the knowledge gained from Case B indicating that Google Cloud having benefits over AWS in this case, most of the time was used in studying Google Cloud. Three introductory Qwiklabs were executed, giving some educating hands-on experience of GCP, virtual machines inside, and cloud shell.

The time taken: 5 days. Compared to the estimation, additional time was taken by the learning tasks and experimentation, which may not be necessary. However, this is information that may end up being useful in later, and accurately identifying superfluous information is not an easy task.

Pre2 Read Google Whitepaper, Zhao & Zhou, and more

Due to the topic of this thesis, a lot more reading was done than prescribed in the task description. The one-day estimation should, however, be an appropriate estimation for the time required.

Pre3 Identify high-level targets

The high-level targets were discussed and identified (see 4.3). The targets were the development environment, testing environment and production environment. Additionally, a potentially favourable temporary environment for additional quick testing deployments was recognized.

The time taken: 1 day.

Pre4 Identify the requirements of the targets

The high-level requirements for the targets were identified. Not surprisingly, considering the description of the system, there are memory and CPU concerns for the production environment, latency requirements between DB and services. Along with requirements for availability and backups. On the other hand, the test environment does not have as high requirements, but the logical layout needs to match the production one. Development environment requirements are not high.

The time taken: 1 day. Due to the identified high level targets being similar to each other, the three identified environments were handled in a day.

Pre5 Create a case for migration

The situation was considered in the light of the current costs and deployments, and the end result in this case was that the current hosting for production environment and testing environment are both cost-effective and have enough resources, so that for now, at least for a year, moving the main components to the cloud would not make sense. Building short-lived testing environments into the cloud was nevertheless identified as a beneficial target. There were some production services identified which could be run in the cloud: Login service, web socket service.

A driver for change, potentially indicating migration, was identified as some requirements for an improved build process, including some automatic tests applied. Another is the monitoring features of built-in cloud environments.

The time taken: 0.5 day

Pre6 Identify knowledge, expertise

This task was executed before the existence of this tool. In this case, the generic level of knowledge and expertise was readily available, but the harder part was estimating the availability of knowledgeable resources for the migration. Moreover, this question is more relevant in cases where there is not someone writing an MSc thesis about it.

The time taken: 1 hour. This will most likely be higher outside this work, and especially in smaller organizations the availability of the experts may prove to be the harder question than identifying the knowledge, as shown by case Cometa and Case A.

Pre7 Identify gaps in knowledge and expertise

This task, like Pre6, was executed before the existence of this tool, with the information readily available.

The time taken: 1 hour. This will most likely be higher outside this work.

Pre8 Go through AWS CART (or similar)

AWS CART was gone through as an exercise to evaluate the validity as a step for the tool. The questions are sensible and direct to the proper direction for a successful migration. As most of the knowledge was already researched for this case, answering was quick, which may not be the norm.

The time taken: 1 hour. This will most likely be higher outside this work.

Pre9 Plan and acquire missing information, required expertise

The need for more accurate CPU and memory requirements was identified, as the resources currently available, and more than adequate, would most likely not be needed in their entirety. More expertise on containerization was recognized as a need for launching quick test environments. Coarse estimations sufficient for continued planning were made

The time taken: 1 day

4.5.2 Planning

Pla1 Identify data and requirements

The environment is very data-driven, as described in the software architecture section 4.2. Thus, there is a lot of various data, but it is well defined, with well-known data architecture, existing primarily in the databases.

The time taken: 0.5 day. Again, this is probably higher in many cases, at least if the documentation is not up to date.

Pla2 Identify features of the current architecture, environment

In this case, a production server migration had been executed by Cometa within a year. Thus most of the information was readily available, even if some specifics had not yet been documented.

The time taken: 1 day

Pla3 Identify migratable systems and their components

As discussed above, the production system and testing system will not be migrated at this point entirely. The system identified for potential migration is the development system, and its components, fewer than contained in the production system. One of the targets here is the improved potential for load testing. Three components from the production system were also included for continued evaluation.

The time taken: 1 day

Pla4 Determine migration strategy/strategies

The primary strategy for the development system is lift-and-shift in the beginning. After this, the migration will continue towards more cloud-native solutions, along with faster and more frequent deployments with containers. The production system components selected will be rehosted, lifted and shifted, and then likewise refactored to containers.

The time taken: 1.5 days

Pla5 Estimate the work needed for migration

The services identified for migration from the production system require some refactoring for more independent operation. Additionally, the use of containers has been considered, but not yet implemented. Thus taking advantage of Docker or any other container solution will require acquiring additional knowledge and experience, and modifications to the deployment procedures.

The development environment migration consists of three main work items. The first is the lift and shift of the environment, the second is improving the deployment pipe for easier deployments, and after this, the third item is allowing for fine-grained, multiple, environment deployments taking advantage of the ease of procuring and shutting down instances in the cloud. These tasks will also provide information on how the environments and their deployments should evolve in the future.

One of the tasks around this point is to analyse the requirements of the components in more detail if the Pre4 or Pre9 tasks do not cover the requirements with adequate detail for estimating the costs of the later steps

The time taken: 1.5 days.

Pla6 Cloud provider selection

Based on the requirements, any of the three considered major cloud providers offer required features. Based on pricing, previous knowledge and data centre locations, the current selection is Google Cloud. Vendor lock-in was considered to be a non-issue, as no specialized services, unique to any single provider, are required.

Additionally, Google Cloud offers the use of preemptible compute engines, which do not promise uninterruptible operation but cost significantly less than regular machine types. These are suitable for testing environment. [31]

The time taken: 0.5 day. Time had already been used outside the case in researching the providers, pricing, offerings. Otherwise would be closer to the tool’s 3 days.

Pla7 Translate the requirements and cloud architecture to migration costs

The migration cost of the selected development system and components is a combination of the learning and experimenting costs, the planning and modification costs, and the actual deployment work, and the testing and validation required.

Setting up the framework for utilizing Docker as part of the environment is an implicit requirement for using it, and all together this was estimated to take 7 days. The estimate covers both the production components and development environment. The planning and modification tasks for the production components was estimated to take 3–5 days per component. The development environment lift-and-shift was estimated to take 3–4 days and containerized version additional 5–8 days, the estimation improving after the lift and shift is complete. Final validation step should be finished in a single day.

The time taken: 1 day

Pla8 Based on the generated knowledge, calculate running costs in the cloud

Using the above mentioned (3.3.4) strategy of running the initial workload with more than enough resources, the testing environment can be run on an n1-standard-8 compute engine, with 375GB local SSD. As preemptible, running 6 days a week, in Finland, the cost is estimated to be EUR 70.53 a month, with a significant amount of the cost coming from the SSD. Google Cloud SQL - PostgreSQL with db-pg-g1-small adds EUR 30.96. Together with network costs this cost all together EUR ~110 / month. It was estimated that optimizing the resources and the running times could bring the price to EUR 40 / month. Note: To be on the safe side, decisions on the cost/benefit calculations should use value closer to the EUR 110.

The calculations for the transfer costs for Webinar solutions were described in 4.3.2, and at the moment, unless the solution is modified to use less traffic, the costs are prohibitive. This could change due to pricing or alternative decisions on routing the network traffic.

The time taken: 1 day. This is a task that could take more time than estimated, depending on the information at hand.

Tasks not (yet) completed:

Pla9 Create a road map

The Pla7 above gives us an estimate for full working days the migration takes. This estimate can be translated to a roadmap when prioritized as considered applicable. However, in this case, at this point, the road map cannot yet be created as there are currently no available resources for executing the migration. The migration will progress on a later date. Thus, the Imp1, Imp2, Up1 and Up2 tasks will not be executed at this point either.

The above estimations are currently estimated to be reasonably accurate for one year, after which they should be revised, unless significant architectural changes occur, requiring an earlier revision.

4.5.3 Implementation

Imp1 Follow the plan

The cost was estimated to be 19–25 workdays in Pla7. The actual cost has not yet been verified.

Imp2 Test that the system is working and fulfils the requirements

The final validation was estimated to take a single day. Note: This is already included in Imp1.

4.5.4 Maintenance and evolution

Mai1 Rightsizing, ++

This has been estimated to take a single day. Note that this can only be done after the system has been running with an appropriate usage load, and the resource usage has been logged.

Mai2 Consider cloud nativity, containers, best practices

This will be executed at a later date.

4.5.5 Total cost

As can be calculated in the generalized notes above, the total time for preliminary work was 12.5 days and planning 8 days, excluding the roadmap. Implementation was not executed at this time, but the cost was estimated to be 19–25 days. These together equals 39.5–45.5 days.

For the sake of comparison, using the default values, calculating with 1 high-level target, and 3 components, results into 9–12+ days for preliminary work, 14–22.5 days for planning and 3.5 days for implementation. All together 26.5–38 days. The values for the tool were selected using some insight into the deployment and the service components, based on the ability to duplicate the services with no effort. As can be seen, calculating with the default values and the actual case results in a similar range of figures. The implementation is an expected exception, as the default values in the tool are for the lift-and-shift migration.

4.6 Review of the cost estimation tool

The time taken for each of the tasks seems to be reasonably well in line with the estimates given in the tool. Differences between the expected work and amount of work encountered in this case were marked in the description of the tasks. Reviewing the descriptions did not cause any major changes, outside some improved wordings which were applied during the course of executing the tasks.

Based on the above thoughts about the tasks themselves, it is no surprise that the total given by the tool seems to be reasonably well in line with the total of the Cometa case. Especially as the implementation of the case included 12 days of Docker related work (see Pla7 in 4.5.2). The framework building, if applicable, would make sense as a separate task as that would make the sources of cost clearer.

The implementation had significant differences to the original estimates. While the stated limitations partially explain them, some better way to offer early estimates for the cost of implementation as early as possible would be helpful. There could be, for example, different default values for different migration types. Nevertheless, the results were satisfactory.

5. ADDITIONAL CASES

These two cases are included as supplementary cases as they provide insight into real-world migration cases and highlight some specific notes from the background and the need for this thesis. The cloud migration cases described here are unrelated to Case Cometa, and the author has been involved in and has observed them before the estimation tool had been created. The cases are anonymized.

This thesis attempts to identify and distil the information gained from these cases and the work and experiences together with research from academia. Hopefully, these will provide some insight for some practitioner in the future.

5.1 Supplementary Case A

The first supporting case involves a cloud migration project of a PHP based backend and the services it uses. The migration was motivated by the need for containerized components for ease of deployment, and issues with the current hosting provider. The selected strategy was lift and shift, with ideas of later improvements bit by bit once the migration steps had been completed.

Additionally, an issue discussed in 2.6 Additional considerations with legacy systems was encountered, as a resource-heavy process was causing problems, and it was impossible to add system resources to alleviate the situation quickly. The situation, once solved, ended up as an item for considering re-architecturing after the lift-and-shift operations were over.

5.1.1 Path to the cloud

The work started with containerization of components and resulted in some target environments running in the cloud. The process was slow, due to a serious lack of resources, and at times due to lack of experience, despite an outside consultant being hired.

5.1.2 Significance to the thesis

This case, along with the following Case B, were the primary motivators for the subject of this thesis. The case A was a qualified success, but when compared to the next case, the progress could have been significantly better, and any educated estimations for the time needed for the migrations would have been instructive.

A cloud expert presented the process of migrating to overpowered compute resources and thus allowing the use of less accurate estimates of required resources until cost optimization during this case. This process is included in section 3.3.4.

5.2 Supplementary Case B

The second supporting case is a cloud migration project where the migration decision was a combination of retaining and re-architecting. The core legacy services were kept, and new functionality was introduced via re-architecting, using cloud-native solutions. The categorization by Zhao and Zhou for this is SaaS with sub-strategy “revising based on SaaS” [37].

5.2.1 History of the software architecture

The architecture of the back end was created as a joint operation by consultants and software development companies. The result was then significantly modified inhouse. The backend is robust but suffers some legacy issues with documentation, understanding of inner workings and clean APIs (see 2.6).

5.2.2 Software architecture

The software architecture before the cloud transformation of the Case B is based on a web front with monolithic backend, which is called through REST-based API (Application Programming Interface). Most of the information in the system is stored in a relational database, but a NoSQL database also exists. Traffic is load-balanced, creating robustness through duplication. Extensive backups are taken.

5.2.3 The reasoning for cloud migration

The primary driver in case was the need to create new features and products and do this quickly and in-house. The target was improving the time-to-market, as it could be argued that the system had some legacy issues, as noted in 5.2.2 Software architecture.

5.2.4 Path to the cloud

The approach taken was revising based on SaaS, as this allowed creating services into the cloud, reducing the dependencies to the backend, and allowing the development resources of the backend to concentrate on specific areas of improvement.

Vendor lock-in considerations

Need for changing cloud vendors was encountered during the case due to unexpected incompatibilities (see 2.5.7 Risks of cloud migration). The original plan, and some work, was executed in the AWS environment, but the final environment was Google Cloud. In this case, as nothing unique to AWS was involved, the change of environment was rather straightforward, which may indicate that the fear of vendor lock-in may not be, at least when sticking to the basic services offered by all cloud vendors, a worrisome issue. On the other hand, exclusive services, such as specific machine learning ones, might be impossible to replace.

5.2.5 Significance to the thesis

The cloud migration was very successful, and while issues were encountered, resulting for example, to changing the cloud provider from AWS to Google Cloud, the migration encountered steady progress with inhouse learning, planning and experimenting. This experience solidified authors belief in both having and increasing in-house knowledge in cloud migrations cases.

6. ANALYSIS AND INTERPRETATION

A significant aspect for consideration in analysing the results is that they are based on a limited case study and are not statistically significant. Due to the minimal amount of research previously published on the costs of migration with a focus on anything else than comparing the running cost of on-premises to running in the cloud [50], the explorative approach makes sense. None of the papers found considered the cost of acquiring expertise, including the Model for Cloud Migration Cost by Antohi, which targets the migration period. An article, with a call for submissions, The Costs of Cloud Migration by Omer Rana comes closest, as human costs and lack of understanding of how the cloud provider operates is mentioned in association with surrendering system management skills to outsourcing. Moreover, the article laments that those skills could have been developed in house. [62] This in-house skill development is exactly what the author of this thesis has seen happening in three separate cases, as the DevOps culture is gaining ground. And especially in Case A there was a need for even an adequate estimation on how long the migration would take, as the number of people available for the migration project was limited. And it is the author's strongly held opinion that this validates the need for this thesis.

Based on the Cometa case, the estimation tool shows promise, but it must be recognized that the tool is somewhat simple. On the positive side, the simplicity has been built with general cases in mind and tries to guide the user in filling in the blanks, as all migrations are unique as discussed in 3.1 Challenges in estimating the migration cost. On the other side, the guidance could be better, and the estimates by default cover more cases. This work could be improved by finding out and verifying, in more detail, the aspects that affect the amount of work and the skills and knowledge needed for successful cloud migration. The trouble is that it is quite easy to list characteristics that might affect the migration cost, for example: number of interfaces, number of components connections, the amount of code, number of commits in version control, and so on. Identifying the ones that actually are significant would take quite a lot of work. Some of the migration models, for example [54], consider the number of components and database changes, but with the caveat that it assumes that all design decisions have already been made. Nevertheless, it could be used as one of the starting points for integrating the amount of implementation work needed as part of the migration cost estimation tool.

Further research could include going through several more cases where cloud migration has either succeeded or attempted and improve the tool with results from these. As is,

the cost in workdays estimated with the tool is somewhat hypothetical. More cases might bring the estimates to the level where the accuracy could be given a reasonable statistical probability. These additional cases might also bring options for cost reductions, best practices, and optimized solutions.

7. CONCLUSIONS

The objective of this thesis was to provide a method for estimating the cost of cloud migration. A novel addition to the existing literature is the inclusion of acquiring knowledge and experience as part of cloud migration. A tool guiding the estimation process was introduced.

The first part of the thesis contains information useful in understanding cloud services and concepts related to cloud and cloud migrations. The collected knowledge and sources used should be helpful for an aspiring practitioner. Concepts introduced included service models, serverless computing, cloud-nativity, the selection of migration targets, and many more.

The second part covers the cost estimation for cloud migration. The challenges in estimating the cost is an important consideration; not a lot has been written about the cost of migration in comparison to the cost of running the migrated environment. This leads to the exploratory cost estimations in this work. The migration cost sources and estimation process is explained. Together the literature reviewed and author's experiences with cloud migrations form the basis for the Migration Cost Estimation Tool.

The third part of the thesis is the case of Cometa Solutions, which is first introduced and then analysed. The MCET is used to estimate the cost of migrating the targets identified in the analysis. The tool produced viable results both when going through the case task by task improving the estimates for the next phases, and also when applying the default values. The default values represent a simple migration, and such can be used for evaluating a minimum cost for cloud migration. Going through the tool, one task at a time is expected to provide useful information in estimating the migration cost in workdays even in more complex migrations. Nevertheless, the work was exploratory and significant areas for further improvement and research were identified in chapter 6.

The estimation of migration costs was the primary purpose of this thesis. In addition to this, the work can also be considered a partial answer to Gholami *et al.*'s question "how would developers grasp a quick and overarching view of what the cloud migration process entails" [7].

REFERENCES

[1] E.F. Boza, C.L. Abad, M. Villavicencio, S. Quimba, J.A. Plaza, Reserved, on demand or serverless: Model-based simulations for cloud budget planning 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), 2017, pp. 1–6.

[2] P. Mell, T. Grance, The NIST definition of cloud computing, National Institute of Standards and Technology, 2011. https://doi.org/10.6028/NIST.SP.800-145.

[3] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, et al., Serverless Computing: Current Trends and Open Problems In: Chaudhary S, Somani G, Buyya R, editors. Research Advances in Cloud Computing, Singapore: Springer Singapore, 2017, pp. 1–20.

[4] S. Kroonenburg, The Next Layer of Abstraction in Cloud Computing is Serverless, Medium 2016. https://read.acloud.guru/iaas-paas-serverless-the-next-big-deal-in-cloud-computing-34b8198c98a2 (accessed 14 January 2020).

[5] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing, ArXiv [CsOS] 2019.

[6] V. Andrikopoulos, T. Binz, F. Leymann, S. Strauch, How to adapt applications for the Cloud environment, Computing vol.95, no.6, 2013, pp. 493–535.

[7] M.F. Gholami, F. Daneshgar, G. Low, G. Beydoun, Cloud migration process—A survey, evaluation framework, and open challenges, J Syst Softw vol.120, 2016, pp. 31–69.

[8] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, et al., A view of cloud computing, Commun ACM vol.53, no.4, 2010, pp. 50–58.

[9] T. Khanye, J. Ophoff, K. Johnston, Issues in Migrating Legacy Systems to the Cloud, 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence) 2018. https://doi.org/10.1109/confluence.2018.8443029.

[10] Cloud Pricing Comparison in 2020, ParkMyCloud n.d. https://www.parkmycloud.com/cloud-pricing-comparison/ (accessed 21 January 2020).

[11] Pricing | Cloud Spanner | Google Cloud, Google Cloud n.d. https://cloud.google.com/spanner/pricing (accessed 29 January 2020).

[12] Network pricing | Compute Engine Documentation | Google Cloud, Google Cloud n.d. https://cloud.google.com/compute/network-pricing/ (accessed 23 January 2020).

[13] EC2 Instance Pricing – Amazon Web Services (AWS), Amazon Web Services, Inc n.d. https://aws.amazon.com/ec2/pricing/on-demand/ (accessed 23 January 2020).

[14] Azure Free Account FAQ | Microsoft Azure n.d. https://azure.microsoft.com/en-us/free/free-account-faq/ (accessed 23 October 2019).

[15] AWS Lambda – Pricing, Amazon Web Services, Inc n.d. https://aws.amazon.com/lambda/pricing/ (accessed 28 January 2020).

[16] D. Gannon, R. Barga, N. Sundaresan, Cloud-Native Applications, IEEE Cloud Computing vol.4, no.5, 2017, pp. 16–21.

[17] D.S. Linthicum, Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between, IEEE Cloud Computing vol.4, no.5, 2017, pp. 12–14. https://doi.org/10.1109/mcc.2017.4250932.

[18] N. Kratzke, P.-C. Quint, Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study, J Syst Softw vol.126, 2017, pp. 1–16.

[19] C.J. Guo, W. Sun, Y. Huang, Z.H. Wang, B. Gao, A Framework for Native Multi-Tenancy Application Development and Management The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007), 2007, pp. 551–558.

[20] M.A. Chauhan, M.A. Babar, Towards Process Support for Migrating Applications to Cloud Computing 2012 International Conference on Cloud and Service Computing, 2012, pp. 80–87.

[21] M.A. Babar, M.A. Chauhan, A tale of migration to cloud computing for sharing experiences and observations Proceedings of the 2nd international workshop on software engineering for cloud computing, 2011, pp. 50–56.

[22] P. Jamshidi, A. Ahmad, C. Pahl, Cloud Migration Research: A Systematic Review, IEEE Transactions on Cloud Computing vol.1, no.2, 2013, pp. 142–157.

[23] Google Cloud migration whitepaper now available | Google Cloud Blog, Google Cloud Blog n.d. https://cloud.google.com/blog/products/cloud-migration/4-steps-to-a-successful-cloud-migration (accessed 11 December 2019).

[24] D. Bernstein, Containers and Cloud: From LXC to Docker to Kubernetes, IEEE Cloud Computing vol.1, no.3, 2014, pp. 81–84.

[25] T. Krazit, Amazon Web Services now fully supports Kubernetes with the general release of Amazon EKS, GeekWire 2018. https://www.geekwire.com/2018/amazon-web-services-now-fully-supports-kubernetes-general-release-amazon-eks/ (accessed 22 October 2019).

[26] A. Balalaie, A. Heydarnoori, P. Jamshidi, Migrating to Cloud-Native Architectures Using Microservices: An Experience Report Advances in Service-Oriented and Cloud Computing, Springer International Publishing, 2016, pp. 201–215.

[27] D. Taibi, V. Lenarduzzi, C. Pahl, Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation, IEEE Cloud Computing 2017.

[28] Free Tier FAQs, AWS Free Tier n.d. https://aws.amazon.com/free/free-tier-faqs/ (accessed 23 October 2019).

[29] GCP Free Tier | Google Cloud Platform Free Tier | Google Cloud, Google Cloud n.d. https://cloud.google.com/free/docs/gcp-free-tier (accessed 23 October 2019).

[30] Qwiklabs, Qwiklabs - Hands-On Cloud Training, Qwiklabs n.d. https://www.qwiklabs.com/ (accessed 10 December 2019).

[31] Qwiklabs, Kubernetes in the Google Cloud | Qwiklabs, Qwiklabs n.d. https://www.qwiklabs.com/quests/29 (accessed 10 December 2019).

[32] Overview - CircleCI n.d. https://circleci.com/docs/2.0/about-circleci/ (accessed 10 December 2019).

[33] M. Hajjat, X. Sun, Y.-W.E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, et al., Cloudward bound, Proceedings of the ACM SIGCOMM 2010 Conference on SIGCOMM - SIGCOMM '10 2010. https://doi.org/10.1145/1851182.1851212.

[34] J. Woods, Five options for migrating applications to the cloud: Rehost refactor revise rebuild or replace Gartner The Future of IT Conference, 2011.

[35] S. Orban, 6 Strategies for Migrating Applications to the Cloud, Medium 2016. https://medium.com/aws-enterprise-collection/6-strategies-for-migrating-applications-to-the-cloud-eb4e85c412b4 (accessed 8 January 2020).

[36] AWS Professional Services, AWS Migration Whitepaper March2018.

[37] J.-F. Zhao, J.-T. Zhou, Strategies and Methods for Cloud Migration, Int J Autom Comput vol.11, no.2, 2014, pp. 143–152.

[38] Q.H. Vu, R. Asal, Legacy Application Migration to the Cloud: Practicability and Methodology 2012 IEEE Eighth World Congress on Services, 2012, pp. 270–277.

[39] O. Ruutiainen, Serverless-arkkitehtuurin hyödyntäminen ohjelmistoprojektissa. Tampereen Teknillinen Yliopisto, n.d.

[40] Your favorite languages, now on Google App Engine | Google Cloud Blog, Google Cloud Blog n.d. https://cloud.google.com/blog/products/gcp/your-favorite-languages-now-on-google-app-engine (accessed 9 December 2019).

[41] G.C. Whitepaper, CIO's Guide to Application Migration 2019.

[42] Cloud Migration Strategies | Microsoft Azure n.d. https://azure.microsoft.com/en-us/migration/migration-journey/ (accessed 22 January 2020).

[43] J. Opara-Martins, R. Sahandi, Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective, Journal of Cloud 2016.

[44] M.F. Gholami, F. Daneshgar, G. Beydoun, F. Rabhi, Challenges in migrating legacy software systems to the cloud—an empirical study, Inf Syst vol.67, 2017, pp. 100–113.

[45] D. Linthicum, Why cloud computing projects fail? 2012. https://www.slideshare.net/Linthicum/why-cloud-computing-projects-fail.

[46] M.L. Brodie, M. Stonebraker, Legacy Information Systems Migration: Gateways, Interfaces, and the Incremental Approach, Morgan Kaufmann Publishers Inc., 1995.

[47] J. Bisbal, D. Lawless, Bing Wu, J. Grimson, Legacy information systems: issues and directions, IEEE Softw vol.16, no.5, 1999, pp. 103–111.

[48] J. Bisbal, D. Lawless, Bing Wu, J. Grimson, V. Wade, R. Richardson, et al., An overview of legacy information system migration Proceedings of Joint 4th International Computer Science Conference and 4th Asia Pacific Software Engineering Conference, 1997, pp. 529–530.

[49] Lei Wu, H. Sahraoui, P. Valtchev, Coping with legacy system migration complexity 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), 2005, pp. 600–609.

[50] T. Antohi, Model for Cloud Migration Cost, 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom) 2019. https://doi.org/10.1109/cscloud/edgecom.2019.00014.

[51] AWS Cloud Adoption Readiness Tool (CART) n.d. https://cloudreadiness.amazonaws.com/#/cart/assessment (accessed 11 December 2019).

[52] Azure Database Migration Service | Microsoft Azure n.d. https://azure.microsoft.com/en-us/services/database-migration/ (accessed 31 January 2020).

[53] Migration from MySQL to Cloud SQL | Solutions | Google Cloud, Google Cloud n.d. https://cloud.google.com/solutions/migrating-mysql-to-cloudsql-concept/ (accessed 31 January 2020).

[54] V.T.K. Tran, K. Lee, A. Fekete, A. Liu, J. Keung, Size Estimation of Cloud Migration Projects with Cloud Migration Point (CMP) 2011 International Symposium on Empirical Software Engineering and Measurement, 2011, pp. 265–274.

[55] D. Huether, The Definition of Done, LeadingAgile 2017. https://www.leadingagile.com/2017/02/definition-of-done/ (accessed 16 December 2019).

[56] Cost Optimization | EC2 Right Sizing | AWS Solutions, Amazon Web Services, Inc n.d. https://aws.amazon.com/solutions/cost-optimization-ec2-right-sizing/ (accessed 23 January 2020).

[57] Yritys – Cometa Solutions n.d. https://cometasolutions.fi/cometa/yritys/ (accessed 29 October 2019).

[58] Ahjo – Cometa Solutions n.d. https://cometasolutions.fi/cometa/ahjo/ (accessed 29 October 2019).

[59] V.I. Lahdenperä, Yleistettävän web-näkymäarkkitehtuurin suunnittelu ja toteutus mallipohjaiseen ympäristöön 2015.

[60] C. Holmberg, S. Hakansson, G. Eriksson, Web real-time communication use cases and requirements, RFC 7478 2015.

[61] C. Koehncke, 4k video and WebRTC - Chris Kranky, Chris Kranky 2017. https://www.chriskranky.com/4k-video-webrtc/ (accessed 23 January 2020).

[62] O. Rana, The Costs of Cloud Migration, IEEE Cloud Computing vol.1, no.1, 2014, pp. 62–65. https://doi.org/10.1109/mcc.2014.24.