MASTER IN COMPUTER SCIENCE

Cloud migration of legacy applications

Kalisa, Flora

Award date:
2015

Awarding institution:
University of Namur

Université de Namur
Faculty of Computer Science
Academic Year 2014-2015

Cloud migration of legacy applications

Flora KALISA

Supervisor: PhD Philippe THIRAN (Signed for Release Approval - Study Rules art. 40)

A thesis submitted in the partial fulfillment of the requirements
for the degree of Master of Computer Science at the Université of Namur

Abstract

Cloud computing has opened up a new way of doing business by hosting and delivering services over the Internet. This business model has allowed many organizations to save money and time and more importantly to focus on their core business. Today, organizations which have adopted the new paradigm are not looking back (e.g. NASA cloud migration success story). However, despite all those benefits, there are still organizations which are unwilling to migrate their legacy applications in the the cloud due to the lack of proper migration methodology and risks and benefits involved. In this paper, we present the key concepts of cloud computing and legacy code and then, we present the state-of-the-art of migrating a legacy application in the cloud by focusing on five frameworks : ARTIST, Cloud-RMM, REMICS, CloudMIG and Legacy-to-Cloud Migration Horseshoe framework. This will allow to suggest some recommendations with the goal of convincing more organizations to adopt this paradigm.

Keywords : cloud computing, legacy application, legacy-to-cloud, cloud migration, application modernization, recommendations

Acknowledgements

Foremost, I would like to express my gratitude to my supervisor PhD Philippe THIRAN for the support of my thesis. His guidance helped me in my research and writing of this thesis.

Besides my supervisor, I would like to thank the Université of Namur for giving me such opportunity to broaden my knowledge and to further my career. My sincere thanks also goes to the Vice Dean Laurent SCHUMACHER, the secretary Benjamine LURQUIN, the teachers and the assistants.

Last but not the least, I would like to dedicate this thesis to my late father.

Table of Contents

List of figures

List of tables

Acronyms

ACID: Atomicity, Consistency, Isolation, Durability
ADM: Architecture-Driven Modernization
API: Application Programming Interface
ARTIST: Advanced software-based seRvice provisioning and migraTIon of legacy
AWS: Amazon Web Services
BASE: Basically Available, Soft state, Eventually consistent
BFT: Business Feasibility Analysis Tool
CAPEX: Capital expenditure
Cloud-RMM: Cloud Reference Migration Mode
CPU: Central Processing Unit
DB: Database
DBMS: Database Management System
DBFE: Database Forward Engineering
DBRE: Database Reverse Engineering
DDL: Data Description Language
FE: Forward Engineering
FE MT: Forward Engineering Model Transformation
HA: High Availability
IaaS: Infrastructure as a Service
ILS: Information Legacy System
IT: Information Technology
ITIL: Information Technology Infrastructure Library
I/O: Input-Output
MDA: Model-Driven Architecture
MAT: Maturity Assessment Tool
NA: Not Applicable
NASA: National Aeronautics and Space Administration
NIST: National Institute of Standards and Technology
OMG: Object Management Group
OS: Operating system
P2P: Peer-to-peer
PaaS: Platform as a Service
PDM: Platform Description Model
PIM: Platform Independent Model
PSM: Platform Specific Model
QoS: Quality of Service
RE: Reverse Engineering
RE MT: Reverse Engineering Model Transformation
REMICS: Reuse and Migration of legacy applications to interoperable Cloud Services

ROI: Return on investment
SDLC: Software Development Life Cycle
SaaS: Software as a Service
SOA: Service-Oriented Architecture
SQL: Structured Query Language
TCO: Total Cost of Ownership
UML: Unified Modeling Language
VM: Virtual Machine

Chapter 1

1. Introduction

Cloud computing, often referred to as simply "the cloud", is a delivery model of on-demand computing resources (e.g. software, hardware, service, etc.) located in datacenters and available over internet. This models has been adopted at an astonishing pace in the marketplace. However, legacy applications are still lagging behind because of their complexity or the use of "old" technologies, thus they are not ready to be moved to the cloud: they are not cloud-enabled.

Legacy applications play an important role in many organizations, because these organizations are still relying on them as they support their core businesses and most importantly, they are often the ones that generate revenues. Today, cloud is happening and legacy applications should take advantages of those benefits such as almost zero upfront infrastructure investment, infinite just-in-time infrastructure, effective resource utilization, reduced time to market and usage-based costing [10].

Organizations concerned by cloud migration, must develop a vision for the future and they must transform themselves to realize that vision. That same vision should be then translated in motivation. The main reasons that motivate organizations in adopting cloud computing are:

- Cost saving [25]: organizations are wasting money and time that can be invested in something else.

- Cost flexibility [9]: Traditional environments do cost a fixed price more often, they are designed to support the worst scenario which is costly.

- Business agility [9]: In traditional environments, IT resources are not acquired and deployed quickly. This means that organizations cannot innovate, introduce new products and services, enter new markets, and adapt to changing circumstances as quickly as they wish. Organizations would like to reduce time to market.

- Business continuity [9]: Organizations want to address the growing demand of traffic and they do not want to get disrupted by unexpected events such as cloud provider bankrupt, fires, floods, and hurricanes that could harm the organization.

- Opportunistic business strategy [12]: Organizations want to remain competitive and meet business objectives. They want to create future value through.

The decision of migrating an application to cloud is complex because due to possibly conflicting factors, such as governance, cost, skills, performance, security and legal obligations [5]. Besides that, there is always confusion for the organization over which cloud to adopt. In addition, legacy applications are often tight coupled to the environments on which they are running and makes the migration process to the cloud even harder.

Migration can be defined as a process of moving to new hardware, new software or both [14] and in particular case of cloud computing, the environment within which the system operates is in cloud. Furthermore, cloud migration of legacy applications requires to transform legacy applications into cloud-enabled application so that those applications can take full advantage of their new environment.

The process of migrating into cloud has to be carefully implemented since IT plays nowadays an important role in business and a failed migration can undermine the reputation and business of the concerned organization. The way that organizations are currently moving to the cloud as once said Jason Bloomberg is by taking the plate of spaghetti and physically putting it in the cloud [32] which is not appropriate because legacy applications that is not engineered to operate in the cloud , will not yield the hyped benefits of the cloud.

In order to migrate legacy applications to cloud, we must first understand the concept of cloud computing and legacy applications. From those two preconditions, we can analyze methodologies that would allow to transform legacy applications into cloud-enabled applications also known as modern applications.

The rest of this chapter will be organized as follows: chapter 2, presents the cloud overview details, chapter 3 gives a broader definition of legacy application, chapter 4 will illustrate the State-of-the-Research of migrating legacy applications in the cloud with a focus on five frameworks: ARTIST, Cloud-RMM, REMICS, CloudMIG and Legacy-to-Cloud Migration Horseshoe framework, chapter 5 shows the main challenges and draws some recommendations. Finally, section 6 offers some conclusions and outlines the future work.

Chapter 2

2. Cloud Computing

2.1. Definition

There has been confusions over the term cloud computing because it has been overused especially in marketing, this has led to some doubts about the true meaning of this business model and more importantly, cloud computing, unlike other technical terms, is not a new technology, but rather a new business model that puts together a set of existing technology concepts. Besides, cloud ecosystem is still evolving[45] which adds even more confusion.

Similar concepts related to cloud computing are:

1- Grid Computing. A distributed system that puts together networked resources to reach a common goal (e.g. computation). Cloud computing is similar to computing paradigm because it employs distributed resources to achieve application-level objectives [3].

2- Mainframe computer. A powerful computers used for large-scale computing purposes that require greater availability and security [18]. Cloud computing is similar to mainframe computers thanks to time-sharing concept, where resources are shared among many users.

3- Utility Computing. Utility computing is business model where consumers pay cloud providers based on usage [3]. Cloud computing is similar to utility computing in sense that consumers are charged based on resources usage. This characteristic is known in cloud ecosystem as pay-as-you-go, pay-per-use or pay-as-you-grow.

4- Peer-to-peer. A distributed architecture with equivalent peers (participants) and connectors among them. Peers are both suppliers and consumers of resources [18]. Cloud computing is similar to peer-to-peer since it shares many of the characteristic of other P2P systems developed for file sharing and content distribution.

5- Virtualization. Virtualization allows to have multiple operating systems on a single physical machine, so they can share underlying hardware resources. Virtualization abstracts physical hardware from virtualized resources [3]. Cloud computing is similar to virtualization because virtualization is the main enabling technology for cloud computing.

Cloud computing benefits from the concepts without having an expertise with them and allows to reduce costs by helping clients to focus more on their core business.

In this paper we will adopt the definition of the United States National Institute of Standards and Technology (NIST) [20] which provides a more complete definition of cloud computing. The NIST defines c*loud computing as model for enabling service users to have ubiquitous, convenient and on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services), that can be rapidly provisioned and released with minimal management effort or service-provider interaction.*

2.2. Cloud actors

There are many actors in cloud computing world but this paper will focus in roles which can be encountered while migrating applications in the cloud:

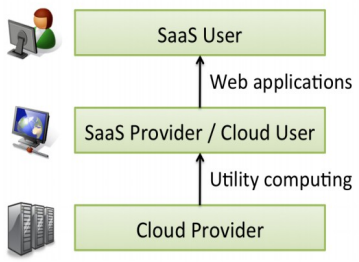

Figure 1: Cloud actors [35]

1- SaaS User. Also known as SaaS consumer is the end user of the application which is hosted in the cloud. This end user can be either a company or a person. The service can be free of charge or monthly or annual subscription. SaaS applications are made accessible via a network.

2- SaaS Provider. He is a supplier to SaaS user to whom he provides an application and he is client to cloud provider to whom he is renting cloud resources. The SaaS provider uses the tools and resources provided by cloud provider to develop, test, deploy, and manage the operation of the application hosted in a cloud.

3- Cloud Provider. Known as infrastructure provider, he is a person or an organization or an entity responsible for making a service available to cloud consumer. The cloud provider has its own datacenters and provides cloud resources (network, storage, tools, etc.) to the SaaS provider. Some big companies may play many roles; (e.g. Google). The relation between SaaS provider and cloud provider is known as utility computing.

4- Cloud Consumer. Known as service owner. A cloud consumer is an organization or a person that develops and manages the applications. Cloud consumer uses service from cloud provider therefore he has a business relationship with him.

2.3. Cloud computing essential characteristics

Cloud computing has many characteristics, however the main characteristics defined by NIST are [19]:

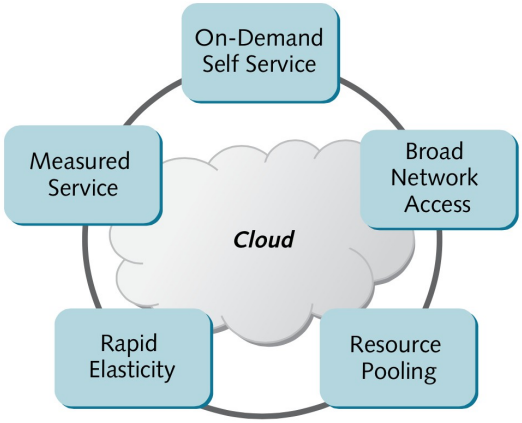

Figure 2: The essential characteristics of cloud computing [9]

1- On-demand self-service. *C*omputing capabilities (e.g. server) available to cloud consumer are provisioned without human intervention. The advantage of on-demand self-service is to lower the operational cost because the human intervention is not needed as the allocation of the resources is dynamic. However, the cloud consumer cannot increase his resources indefinitely. Tools in form of API and applications are provided by the cloud provider to increase resources.

2- Broad network access. The SaaS user access the service through the network, generally by Internet. The advantage of broad network access is that cloud services can be accessed by anyone, anywhere on the globe, using internet and a variety of devices (e.g. smartphone).

3- Resource pooling. Resources are shared between multiple cloud consumers with dynamic allocation. For cloud consumer, this means a high Quality of Service (QoS) at low cost because the resources can be shared at any level: infrastructure, platform or software. Cloud consumers may be able to specify location at their resources at higher level of abstraction (e.g. country, state, or datacenter)

4- Rapid elasticity. The cloud consumer can provision and release its capabilities, in some cases automatically in order to correspond to the real demand. In other words, rapid elasticity means the ability to have a flexible computing service which can expand or contract in line with business demand (See Figure 10).

The elasticity is an important characteristic in cloud in sense that clients do not have to bother about how the application will behave depending on the workload (static, periodic, once-in-a-lifetime, unpredictable, continuously changing workload [1]).This feature of elasticity is known as linear scaling [9] where the performance experience for one of a thousand users is the same as for a single user**.**

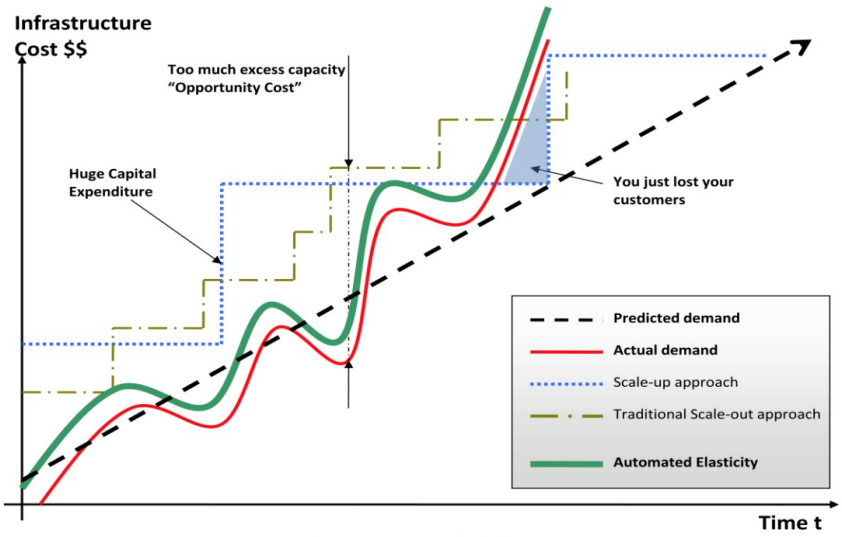

Figure 3: Automated Elasticity [10]

Another feature of elasticity is pay-as-you-go model, where the user pays for consumption of the service on the basis of the resource units consumed [9] (see Figure 10). This also means that the clients are not paying for unused resources therefore they can save money. However, if the service has to be charged as you go, it is obvious that the service has to be measured. Elasticity is the main enabler of pay-as-you-go model.

5- Measured service (Monitoring). Monitoring allows to control resources usage. This characteristic is very important because the cloud services introduce an abstraction which limit control for the cloud consumer. Therefore, the cloud consumer needs tools for resources monitoring in order to know what is happening and check if he is really paying for what is being used. Cloud offers monitoring tools at different levels (network, application, etc.)

2.4. Cloud deployment models

Cloud computing offers four different deployment models [19]:

1- Private cloud. Private cloud offers exclusive use to a single organization. This cloud infrastructure may be owned, managed, and operated by the organization, a third party, or some

combination of them, and it may exist on or off premises [19]. The private cloud offers the control over the infrastructure and data. However, an up-front investment is needed to setup the environment. There is a misunderstanding that private clouds do offer security, this has to be nuanced though. It offers more control but if the right security in not implemented, the private cloud cannot help out.

Example: OpenShift Enterprise is an on-premises, private Platform as a Service (PaaS) solution.

2- Public cloud. Cloud infrastructures are provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider [19]. The advantage of public cloud is that the cloud consumer is paying for CPU (Central Processing Unit) hour's usage, gigabytes months stored but he is not paying for cooling or for management and maintenance expenses even if we can say it is indirectly included in the pricing model.

Example: OpenShift Online is a public Platform as a Service solution

3- Hybrid cloud. Hybrid cloud is a composition of two or more clouds (private, community, or public). Hybrid cloud is what clients often implement these days [5] and the most popular combination is using private and public cloud. This allows cloud consumer to avoid some of the main issues encountered while migrating to either public cloud (e.g. data security, data sovereignty, trust) or private (e.g. up-front investment). However, hybrid cloud faces challenges such as workload portability, network interoperability and secure interconnection between the two clouds.

4- Community cloud. A community of consumers share cloud infrastructure and have shared concerns [45] (e.g. mission, security requirements, policy, or compliance considerations). It may be owned, managed, and operated by one or more organizations in the community, a third party, or some combination of them, and it may exist on or off premises [19]. Community cloud helps community members to share the expenses which can be more interesting than set up a private cloud.

Example: salesforce community cloud

2.5. Cloud service models

Cloud computing offers four different service models:

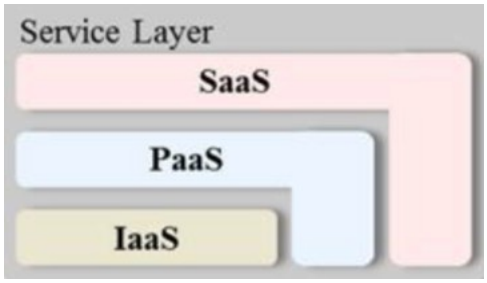

Figure 4: Cloud service models [19]

1- Software as a Service (SaaS). A Consumer uses provider's applications running on a cloud infrastructure[19]. SaaS removes the need to manage both the application and the infrastructure on which the application is deployed. A SaaS provider gives subscribers access to both resources and applications. In SaaS agreement, the client has the least control over the cloud compared to other service models. One way to achieve the implementation of SaaS is Service-Oriented Architecture (SOA) which promotes services reusability, granularity, loose coupling, interoperability and encapsulation and etc. As for roles, a SaaS consumer uses the application and a SaaS provider installs, manages and maintains applications on cloud infrastructure (see Figure 5).

Example: Google docs, MS Office on demand

2- Platform as a Service (PaaS). A platform is provided to the consumer, so that he can deploy his application [19]. The drawback of this service is that organizations have to use tools supported by the provider. The advantage of a PaaS is that a lot of the provisioning is done for you compared to Infrastructure as a Service (IaaS), so it is easier, but it is less flexible than IaaS. The cloud consumer gets to control the deployed applications and possibly configuration settings for the application-hosting environment. As for roles, the PaaS consumer develops, tests, deploys, and manages applications hosted in a cloud system and a PaaS provider provisions and manages cloud infrastructure and middleware for the platform consumers; provides development, deployment, and administration tools to platform consumers (see Figure 5).

Example: Google's AppEngine, Google Compute Engine

3- Infrastructure as a Service (IaaS). The cloud consumer gets access to flexible computing storage infrastructure, operating systems, deployed applications and possibly limited control of selected networking components. One way to achieve the implementation of IaaS is visualization. As for roles, IaaS consumer creates, installs, manages, and monitors services for IT infrastructure operations and a IaaS provider provisions and manages the physical

processing, storage, networking, and the hosting environment and cloud infrastructure for IaaS consumers (see Figure 5).

Example: Amazon Web Services AWS (EC2, S3), Eucalyptus, Rightscale, Microsoft Azure.

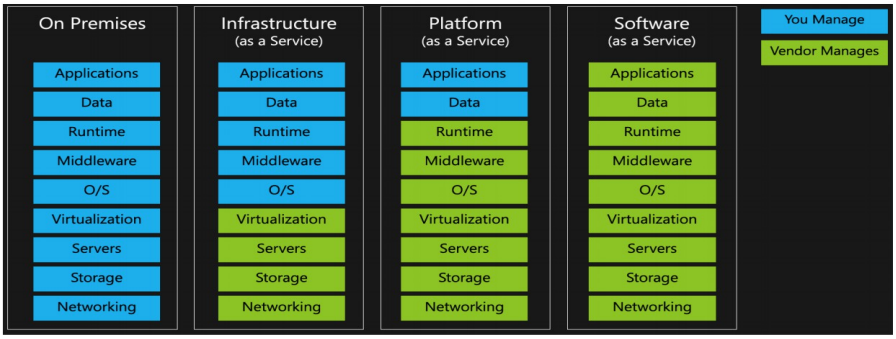

Figure 5: Responsibilities with regards to service models

In the end, we can say that SaaS is software hosted for organizations thus organizations are using the applications offered by others. Organizations could even be using the applications that happen to be hosted using a PaaS or IaaS platform. Whereas, PaaS and IaaS mean organizations run their business respectively in the cloud hosted platforms and cloud hosted infrastructure.

2.6. Differences between cloud and traditional environments

Differences exist between cloud and traditional environments on economic and technical perspectives.

On economic perspective, we differentiate three elements:

1- Time. While it can take days or weeks to setup the traditional environment due to installation and configuration, it only takes minutes or hours to setup cloud environment [9]. However, this also means the necessary tools that allow legacy application to be diagnosed properly exist. Cloud computing introduces an abstraction layer which aim at providing the ability to easily develop and deploy services into clouds but also it introduces a black box for the cloud consumer since there are parts of the cloud architecture that are no longer directly accessible to the cloud consumer specially in case of public cloud.

2- Capital Expenditure. Cloud computing can be described as *converting capital expenses to operating expenses*[18] thanks to the use of pay-as-you-go model which eliminate upfront investment. Furthermore, a good assessment will allow organizations to know if there is added

value to migrate in the cloud. For applications that need to run 24/7, it might be more expensive to migrate them in the cloud than keeping them on premise because of high total cost of ownership (TCO), mainly resulting from migration costs [5]. The migration cost should not be ignored in capital expenditure calculation.

3- Economies of scale. Depending on the deployment models, organizations can save money on hardware, software therefore taking advantage of improved productivity. Besides, with cloud models, there is need to duplicate environments. However organization wanting to migrate their legacy applications should first make the change necessary in order to fully take advantage of the cloud environment. A spaghetti code cannot be moved into the cloud and expect from it economies of scale.

On technical perspective, we also differentiate three elements:

1- Virtualized. Cloud computing environments are usually virtualized, whereas traditional environments include a mix of physical and virtualized infrastructure [9]. Virtualization is about resources sharing. Again performance, efficiency and robustness should be guaranteed for each consumer. The advantages of virtualization are maintainability, cost effectiveness, consolidation, energy and space gain. The ultimate goal of virtualization is the orchestration, where business needs can be defined and executed without human intervention. Virtual Machines (VM) can be scaled by replication, resizing or replacement.

2- Multi-tenancy. Multi-tenancy is a software pattern architecture that allows to effectively use resources if properly applied. Multi-tenancy distinguishes two different consumer types: tenants and users. A tenant is a group of users and tenants can separate cloud consumers too using multi-tenant application [34]. The pattern allows to host multiple tenants across shared resources and it can help in saving costs given that the tenants will share the same application instance. Multi-tenancy implies a need for policy-driven enforcement, segmentation, isolation, governance, service levels, and chargeback/billing models for different consumers[45]. The opposite of multi-tenancy is known as single-tenancy, where each tenant has its own instance of software.

IaaS	PaaS	SaaS
Readily enabled through virtualization	Platforms re-architecting	Applications re-architecting
	Applications re-architecting	

Table 1: Multi-tenancy with service models

Platforms re-architecting is composed of resources and infrastructure while applications re-architecting is composed of storages and applications. Multi-tenant and single tenant are an important notions because they are used in storage and application scalability.

3- Scalability. Scalability allows to dynamically scale up when there is a greater need for additional resources and scaling down when the demand is low. Scalability is among the elements that attract organizations in cloud migrating and it results in increased performance proportionally to the resources added.

The scalability can be achieved through elements below:

(a) Resources scaling. Resources scaling can be done in two different ways:

> **+ Vertical scaling up**. Also known as vertical scaling or scaling up. The main idea is to increase the capacity of individual node through hardware improvements. Practically, vertically scaling is done by adding or removing instance resources, hot or cold swapping or doing multiple configuration.
>
> **+ Horizontal scaling out**. Also known as horizontal scaling or scaling out, increases overall application capacity by adding entire nodes. Practically, horizontal scaling is done by adding or removing new instance.

Resources scaling is possible with cloud provider where it is necessary to access the hardware to make the necessary changes.

(b) Software infrastructures. Infrastructures that can help in scaling cloud environment are:

> **+ Load-balancing**. Load balancing allows to distribute the workload evenly across multiple nodes [27]. The scalability depends on algorithm chosen for the application to perform load-balancing within given resources. As it is said in [29], Load-balancing has at least three major applications: server, firewall and cache. Load-balancing can be implemented by the cloud provider (e.g. Elastic Amazon Load Balancing) or by the SaaS provider (e.g. Resonate, Rainfinity, and Stonebeat).
>
> **+ Auto scaling**. cloud providers offer mechanisms to automatically scale up and down virtual machine capacity based on user defined policy[24] (e.g. AWS auto-scaling). The auto scaling is activated, thanks to the use of triggers, thresholds and performance metrics and it is the property that allows to increase performance proportionally to the resources added without human intervention.

Amazon EC2	Windows Azure	Google App Engine
Based on scalability parameters specified by users	Based on application roles and a configuration file specified by users	Transparent to users

Table 2: Example of auto-scaling activation

+ Network. A scalable network is a network that can work well regardless the amount of users, nodes and data.

+ Container replication. Components are replicated on different containers for better scalability.

+ Snapshots. A snapshot is the state of a system at a particular point in time [60]. Snapshots encourage cloud computing conservationism [35] and allow to rapidly create an instances without starting from scratch.

(c) Software. On software level, an application to be scaled with the following elements:

+ Loose coupling. Loose coupling allows to scale components independently of each other. This approach is also known as Service Oriented Architecture (SOA) which allows to define an architecture which focuses on reusability, granularity, loose coupling, interoperability, encapsulation and etc. Low coupling goes with high cohesion and thanks to those properties applications can easily be maintainable and evolvable. Loose coupling is an important notion in migration process because it can allow to migrate only a part of the application.

+ Software tuning. Software tuning can be defined as a software which runs in good performance under given condition [17] and it can be implemented by the SaaS provider. If the application is not performing well, it is important to identify the bottleneck and then modify the part of the application to remove the bottleneck. Afterwards, measure and take the decision on keeping the changes or not.

+ Multi-tenancy. Multi-tenancy will allow multiple tenants to use the same software. Changes are necessary on different levels. Firstly, the authentication needs to be adapted accordingly and depending on multi-tenancy database approaches, the authentication implementation will differ. Secondly, configuration will allow software customization per tenant. Lastly, software implementation are necessary in term of workflow, layout, configuration, files I/O.

(d) Storage. Storage are used to store data. Data scalability must be ensured through the ability of the system to process increasing amounts of data without undermining overall application availability, performance and throughput. Storage scalability can be achieved on both cloud provider and SaaS provider with elements below:

+ Multi-tenancy. Three approaches exist in managing tenants. Firstly separated databases may be an option if a strong data isolation is needed (e.g. banking and medical applications). Secondly, separate schemas may be an alternative if a moderated isolation is an option, however, the restore of single tenant is complicated. Lastly, shared schema allows to differentiate tenants by giving them an identification in table, however this requires to make changes to queries at application level.

+ Distributed caching. Distributed caching allows to increase scalable performance of applications by first checking if the element exists in memory, if so, it is retrieved from there, otherwise, it is retrieved from the database (e.g. AWS ElastiCache), and many topologies exist with their advantages and disadvantages.

+ NoSQL databases. NoSQL stands for "Not Only SQL" or "Not Relational" and data are stored in a "schema-free" file or data block. NoSQL is DBMS without the "R" from RDBMS and usual it lacks transactions and other sophisticated features. The database implements BASE properties [46] (Basically Available, Soft state, Eventually consistent) and has many advantages such as the ability to scale horizontally over many servers. Different types of NoSQL databases exist as shown below:

NoSQL Database types	Scalability
Key-Value Stores. Data is represented as a collection of key-value pairs where the key is a unique identifier (e.g. Amazon DynamoDB, Oracle NoSQL).	High
Column stores. Databases are designed for storing data tables as sections of columns of data, rather than as rows of data (e.g. Google BigTable).	High
Document stores. Support more complex data than key-value stores**.** Document stores are designed for storing, retrieving, and managing document-oriented information (e.g. MongoDB, CouchDB).	Variable
Graph databases. It uses a graph of nodes with references among them (e.g. Neo4j, OrientDB).	Variable

Table 3: NoSQL database types

+ SQL databases. SQL databases implement ACID properties [46] (Atomicity, Consistency, Isolation, and Durability). They are generally not adapted for cloud scalability, however, they can scale with advanced functionalities such as, firstly, clustering [47] which replicates data among different nodes for the purpose to increase scalable performance. Secondly, partitioning [47] allows to divide large tables on different nodes based on ranges of values known as partition key.

2.7. Synthesis

In this chapter, we have defined and presented cloud computing concepts, its actors, its essential characteristics, deployment and service models. We have highlighted the difference between cloud computing and traditional environment and the benefits offered by cloud computing are undeniable in term of scalability, virtualization and multi-tenancy. Cloud computing can help organizations reduce cost and focus on solving their domain problem. It can be a good investment in migrating legacy applications in the cloud even though this implies a change in roles and responsibilities for organizations.

<table>
<tr><th colspan="5">Application</th></tr>
<tr><td>Scalability</td><td colspan="4">
<table>
<tr><th>Resources</th><th>Infrastructure</th><th>Software</th><th>Storage</th></tr>
<tr><td>Vertical</td><td>Load-balancing</td><td>SOA</td><td>Caching</td></tr>
<tr><td>Horizontal</td><td>Auto scaling</td><td>Multi-tenancy</td><td>Multi-tenancy</td></tr>
<tr><td></td><td>Network</td><td>Tuning</td><td>NoSQL</td></tr>
<tr><td></td><td>Container</td><td></td><td>RDBMS</td></tr>
<tr><td></td><td>Snapshots</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>At cloud consumer's disposal</td><td colspan="4">
<table>
<tr><th colspan="2">IaaS</th><th>PaaS</th><th>SaaS</th></tr>
<tr><td colspan="2">Virtual Machine</td><td>Server
DB
Libraries</td><td>Application</td></tr>
</table>
</td></tr>
<tr><td>Cloud deployment models</td><td>Private cloud</td><td>Public cloud</td><td>Hybrid cloud</td><td>Community cloud</td></tr>
<tr><td>Cloud service models</td><td colspan="4">SaaS, PaaS, IaaS</td></tr>
<tr><td>Cloud essential characteristics</td><td colspan="4">Broad Network Access, Rapid Elasticity, Measured Service, On-Demand Self-Service, Resource Pooling</td></tr>
</table>

Table 4: Cloud computing synthesis

Chapter 3

3. Legacy applications

3.1. Definition

Legacy applications can be defined as *any information system that significantly resists modification and evolution* [23]. Usually, they have poor system design, software architecture or software development. Consequently, they accumulate technical debt which come in the form of the extra effort that has to be done in future development. Legacy applications are any existing project that are difficult to maintain or extend.

Typical, they are large, old, inherited and poorly documented [7]. They are build using old technologies and run on old hardware's and they are difficult to replace because of their wide use. According to [21], they lack tests and they suffer from code tangling (mixing concerns) and code scattering (duplication) because they do not have a proper modularization in their implementation. They are characterized by a tight coupling and a low cohesion.

According to [7], a legacy project is composed of code, data, database, configurations, infrastructures, tools, documentation, and dependencies.

3.2. Legacy applications concerns with regards to scalability

Besides their characteristics, legacy applications have problems that can undermine cloud migration process: Firstly, on resource level, applications lack optimization because they are usually developed to run on their own environment on-premises without taking into consideration resources sharing and optimization and most of the time, resources are added manually with human intervention. Secondly, on infrastructure level, their environments are poor and old therefore they are not optimal. Besides, the scaling is done manually. Thirdly, on application level. They are designed to be single-tenant and they lack cohesion, reusability and are tight coupled. They may be developed using old technologies. Lastly, on database level, their databases are SQL based which are not adapted for scalability unless partitioning and clustering are implemented. They may run on unsupported version of the database in the cloud.

Resources	Infrastructures	Application	Database
Lack of optimization	Old	Old technologies	Old
Manual setup	Poor	Lack of optimization	Lack of optimization
	Manual scaling	Single-tenant	Single-tenant
	Mix of physical and virtualized infrastructures	Tight coupling	RDBMS

		Low cohesion	Lack of distributed caching
		Lack of reusability	Lack of partitioning
			Incompatibility

Table 5: Legacy application concerns with regards to scalability

In the end, legacy applications cannot be migrated as such, they need to go through the necessary changes in order to make them cloud-enabled. The opposite of it will fail to expect from them economies of scale.

3.3. System evolutions

Legacy applications go through four evolutions:

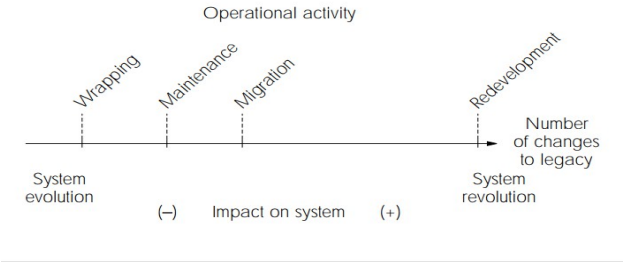

Figure 6: System evolutions [23]

1- Wrapping. This evolution allow to build a new interface on the top of legacy applications therefore it focuses on inputs and outputs of the applications. It is also known as **black-box modernization** technique. Wrapping may be an option if the legacy code is too expensive to rewrite, is relatively small, can be reused, and a fast cost-effective solution is needed [2]. In case wrapping is used in cloud migration, the wrapped application may not take fully advantage of cloud computing. As shown on Figure 6, the wrapping has the least impact on the system evolution.

2- Maintenance. Process in which small modifications are made to correct faults. According to [26], the changes are often bug corrections or small functional enhancements. Maintenance can be expensive, because of the lack of documentation and understanding and tracing faults can be costly and time consuming. Besides, changes on the system are slow and they can have a big impact on the business. Legacy applications are generally in this step. According to [23], maintenance was added on Figure 6, only for completeness, in fact, maintenance is part of

every system's life cycle and if software systems can be maintained within an acceptable budget it is usually not considered as legacy information system evolution.

3- Migration. Process of moving to new hardware, new software or both [14]. Different migration options exist (e.g. re-platform, modernization, etc.). In this study, we are going to focus on modernization which is a technique to rebuild legacy application in a new technology or platform, with same or enhanced functionalities [14], so that the new application is cloud-enabled. Migration can be an alternative when redevelopment is unacceptably risky and wrapping is unsuitable [23]. Modernization complexity comes from understanding legacy application and transforming it in order to take fully advantage of the new environment.

4- Redevelopment. The application is developed from scratch during redevelopment process. It is referred as Big Bang also known as Cold Turkey [23]. This evolution has substantial risk of failure and it should only be considered if the cost of maintenance is higher than cost of rewriting. Redevelopment refers to reengineering approaches even though reengineering is often used as a synonym for migration. It is included in redevelopment category because it requires a thorough understanding of the existing system and thus involves many reengineering activities [23]. Redevelopment strategy is also known as a **white-box modernization** technique. As shown on Figure 6, the impact on the system is much bigger compared to other evolutions because of the risk involved in starting from scratch.

3.4. Migration strategies

The migration evolution can be applied using different strategies according to [13] and they can be classified into two dimensions:

1- Database strategy dimension. It identifies two major components, namely the data and the programs, it distinguishes two strategies based on which component is migrated first [11]:

> **+ Database first strategy**. A new database completely replaces the legacy one, so that their lifespan do not overlap. Two options are available for legacy application: either, it communicates with the new database through wrappers, or it still uses the legacy database, in this case then a synchronization must be established between the old and the new database. [11]. There is cost of introducing synchronization or wrappers. Besides, synchronization can introduce inconsistency between the new and old database.
>
> **+ Database last strategy**. Programs are migrated first to the new environment, thus they are still using legacy database through wrappers. The new application is also using the same wrappers. When all the applications have been converted, the database itself is migrated [11]. Here also, there is a cost of implementing wrappers.

2- Replacement strategy dimension. It concerns the time frame within which the replacement is carried out. It distinguishes two strategies [11]:

> **+ Big bang approach**. A new system replaces the legacy system. According to [11], the substitution is generally carried out in a very short time, typically a few days, so that both systems run with no overlap [11]. The big bang approach is a costly and risky process because if the migrations fails, it is the entire project that fails.
>
> **+ Chicken little approach**. The database and the application are migrated piece by piece [11]. Chicken little is a low risk, thanks to the allocation of small resource in short time on a small part of the legacy application. However this approach introduces overlapping functionalities between the new and old system.

A migration strategy should be based on cost-benefit analysis and it is important to know in the first place because it can help organizations in estimating time and the risk involved during cloud migration process as well as the capital needed. The most appealing choice for legacy application migration is chicken little migration strategy because if the small step fails, it is not the entire project that fails.

3.5. Synthesis

In this chapter, we have presented what is a legacy project and its characteristics. We have presented concerns of legacy application with regards to scalability and the evolution of systems after it has been built. We have noted that a migration strategy is necessary and the choice of it will depend on applications as well as organizations constraints. The legacy applications face many challenges when it comes to migrating them in the cloud but organizations should take the bold move to migrate their applications in order to remain competitive but not necessary at any cost.

Legacy applications running on traditional environment are not generally scalable therefore they need to through the necessary changes to make them cloud-enabled before they are moved into cloud environment. Legacy applications go through four evolutions: wrapping, maintenance, migration and redevelopment. The maintenance is the actual evolution of legacy applications and migration, is the next evolution that will allow legacy applications to be cloud-enabled.

Legacy Application		
Legacy application artifacts	Code, data, database, configurations, infrastructures, tools, documentation, dependencies	
Concerns	**No Technical aspects**	**Technical aspects**

<table>
<tr><td></td><td colspan="4"><table>
<tr><td>Pool documentation
Inherited
Large
Old
Technical debt</td><td>Old
Inflexible code
Untested, untestable code
Difficult to extend and maintain
Single-tenant
Scalability
Resources optimization
Poor infrastructures</td></tr>
</table></td></tr>
<tr><td>Evolutions</td><td>Wrapping</td><td>Maintenance</td><td>Migration</td><td>Redevelopment</td></tr>
<tr><td>Strategies</td><td colspan="4"><table>
<tr><th>Database strategies</th><th>Replacement approach strategies</th></tr>
<tr><td>Database first strategies</td><td>Big bang approach</td></tr>
<tr><td>Database last strategies</td><td>Chicken little approach</td></tr>
</table></td></tr>
</table>

Table 6: Synthesis of legacy applications

Chapter 4

4. Cloud migration of legacy applications

4.1. Prerequisites

The focus of this chapter is about transforming legacy applications in order to make them cloud-enabled so they can leverage cloud environments. It is also about having the same or enhanced functionality between the legacy application and the new application. However, there are types of applications which do not necessary need to be transformed : firstly, embedded applications [12],[37] cannot benefit from the cloud environment therefore their migration into the cloud does not make sense; secondly, cloud-native applications [12], [37] are softwares implemented to work in the cloud therefore they do not need any transformation; lastly, software-intensive applications [12], [37] cannot easily utilize cloud-based environments therefore they represent a higher risk in migrating them.

Depending on service models, organizations should be aware of elements to migrate :

Migration		
IaaS	**PaaS**	**SaaS**
Application	Application	NA
Data and data schema	Data and data schema	
Runtimes (e.g. libraries)		
Middlewares (e.g. Servers, DB, SOA)		
OS		

Table 7: Migration elements per service models

1- On IaaS. Besides migrating applications, data and data schemas. Organizations need first to install Operating System (OS), middlewares, and runtimes elements on the Virtual Machine (VM) in the cloud.

2- On PaaS. Organizations need to deploy their applications, data, data schema on the new platform in the cloud.

3- On SaaS. There is nothing to install because organizations are using applications offered by others in the cloud.

As already mentioned in chapter 2, cloud enabled applications should be designed to be SOA compliant. However, this is not enough to fully take advantage of cloud environments, they also need to be scalable on different levels : resources, infrastructure, application and database.

4.2. Definition

Some terminologies need to be clarified before we go further :

4.2.1. Modeling languages

The Unified Modeling Language (UML) provides modeling concepts to represent software, platform and infrastructure artifacts and also used to model elements below :

(a) Platform Independent Model (PIM). It is a conceptual model and describes the system independently of the platform and technology used.

(b) Platform Specific Model (PSM). The model describes how the system uses the platform and it is used to generate the code for a particular platform.

(c) Platform Dependent Model (PDM).The model contains information for model transformation to a platform. Thus, It allows to transform PIM to PSM and it is specific to a platform.

PIM, PSM and PDM are standard models defined by Object Management Group (OMG).

4.2.2. Transformation languages

1- The Architecture-Driven Modernization (ADM). It is a bottom-up approach that allows to extract architectural models for the the purpose of migration, reusability, improvement, interoperability, refactoring, etc. According to [16] modernization starts when existing practices fail to deliver against business objectives. The ADM is based on Reverse Engineering approach, which can be defined as a process to discover and understand the legacy code without proper documentation by extracting system abstractions and design information from the legacy code. As stated in [41] the process involves software artifacts identification, interaction between artifacts and aggregation to form a more abstract system representations.

2- The Model-Driven Architecture (MDA). It is a top-down approach that allows to develop new systems. The MDA is based on forward engineering approach which can be defined as a process to implement a new system from high-level and Platform Independent Models. According to [6], MDA allows to separate business and application logic from the underlying technology, thanks to the use of PIMs and PSMs. The MDA will automate the transition from PIMs to PSMs, thereafter PSMs and PDMs will be used to generate the new code.

ADM [16] and MDA [6] are transformation models defined by Object Management Group (OMG).

The next section will help to understand, how to transform legacy applications into cloud-enabled applications with the help of different frameworks.

4.3. State-of-the-Research of different frameworks

Regarding cloud migration of legacy applications, there are a number of ongoing projects and process models such as :

ARTIST[40]	Project	Advanced software-based seRvice provisioning and migraTIon of legacy	[P1]
Cloud-RMM[12]	Process model	Cloud Reference Migration Mode	[P2]
REMICS[39]	Project	Reuse and Migration of legacy applications to interoperable Cloud Services	[P3]
CloudMIG[38]	Project	Model-Based Migration of Legacy Software Systems into the Cloud	[P4]
Legacy-to-Cloud Migration Horseshoe framework[36]	Process model	A Framework for Architecture-driven Migration of Legacy Systems to Cloud-enabled Software	[P5]

Table 8: Cloud migration frameworks

These approaches aim at developing a systematic, disciplined and quantifiable process thanks to the use of frameworks, patterns and processes. Systematic means all the processes follow a certain methodology even though tasks can be different from one project to another. Disciplined means there is a plan driven-method aiming at producing quality work with the use of model-driven application engineering. Whereas quantifiable allows to base the decisions on concrete facts.

[P1], [P3], [P4], [P5] are semi-automated which means some tasks are carried out manually and others are being done using suitable tools. [P2] is conceptual model that classifies researches in terms of distinct processes and concerns that cannot be cleanly decomposed from the rest of the system (cross-cutting concerns) [12]. It is a consolidation of around twenty research.

The frameworks are adapted to IaaS and PaaS service models even if software can be served as SaaS to the end user. All the frameworks aim to bring added values in cloud migration process of applications.

	Added values
[P1]	- **Business feasibility** offers means to estimate costs, benefits and operational risks [28]. - **Tool-supported** framework with many open-source tools to ease the process [15]. - The legacy application transformation is **platform-independent** [22]. - **Certification** process guarantees the compliance of migrated application [4].
[P2]	- The framework is very detailed, it is a **characterization framework** [12]. - The framework insists on concerns that cannot be cleanly decomposed from the rest of the system : **cross-cutting concerns** [12]. - The framework is a comparison of systematically selected studies to point out **existing research gaps** [12].
[P3]	- **Agile practice** (Scrum) are used for adaptive, incremental and iterative process [39]. - **Interoperability** is promoted during the migration life-cycle [39]. - **Tool-supported** framework with many custom tools to ease the process [39].
[P4]	- The process is applicable to **object-oriented systems** [38]. - **Tool-supported** framework with only one tool to ease the process [38]. - The tool allows to get **metric reporting** extracted from models [38].
[P5]	- **Incremental migration** of architectures [36]. - Extends the **classical reengineering horseshoe model** where transformation evolves around source code layer, pattern and style layer and architecture layer [36]. - The framework aims to discover, document and apply the **migration process patterns** that enhance the reusability of migration processes [36].

Table 9: Frameworks added values

Cloud migration of legacy applications go through the same processes as traditional migration. By that, we mean that we have Pre-Migration, Migration and Post-Migration processes. However tasks performed are totally different because of legacy applications' characteristics.

4.3.1. Pre-migration

The first process is the pre-migration phase. Instead of gathering business requirements, the information regarding technical feasibility study [P1], [P2], [P3], [P5]; business feasibility study[P1]; decision on cloud provider [P2], [P5]; decision on cloud services[P2]; maturity assessment[P1]; migration requirements [P2], [P3], [P5]; migration strategy [P2], [P5], sub-system to be migrated[P2] are carried out. Some of the frameworks[P4] tend to ignore this phase and they would rather start with the precondition that this phase is favorable to cloud

migration or it was done separately. Not all organizations take all decision in the pre-migration, they can still decide during the next phase (e.g. cloud provider for [P4]).

The pre-migration is important in cloud migration because it allows to know if organizations are ready for the change, if they have the capabilities in terms of compliance, money, resources, knowledge, skills, etc. to undertake such process and if they understand the risk involved .

It is not enough to limit the study to technical aspects as some frameworks already do because the risk can be on on high level such as business or even higher. Because cloud migration process cost money and time, it needs approval from the top level management.

The outcome of the pre-migration phase is a migration plan with decisions on :

Contents	Explications
Go or No-Go	Based on technical and business feasibility studies. If it is No-Go, no need to go further
Cloud migration motivation	Decision based on the organization's vision
Migration analysis	High level architecture analysis from technical feasibility
Security and compliance	Knowing the constraints can help in implementing the right decisions (e.g. narrowing the choices of service and deployment models upfront)
Cloud provider	Decision on deployment model Decision on service model Multi-tenancy supported
Migration strategy	Choice between : Database first strategies Database last strategies Big bang approach Chicken little approach
Component to migrate	Dependencies to the component are isolated and strategies worked out to handle these dependencies.
Technology choice	Choice has to be made (e.g. Java EE, .NET, etc.)
Tools	Which tools can be used in the cloud and which one needs to be built
Create a plan and measure	In order to measure the success of the migration and learn from the experience

Table 10: Migration plan

The Migration plan is a high-level plan and does not exclude to plan on lower levels (e.g. Activity, tasks). The Pre-migration is an important step in cloud migration and organizations should not be underestimate as the success of the whole migration process will depend on it.

4.3.2. Migration

The second process is the migration phase whose input is the migration plan from the previous phase. All the frameworks[P1], [P2], [P3], [P4], [P5] use cloudification as migration type.

Cloudification is a *complete migration of the application where application functionalities are implemented as a composition of services running on the Cloud*[37]. As stated in [37], in addition to any adaptive actions to address possible incompatibilities, it requires data and business logic migration to the cloud. If correctly done with the implementation of scalability, this migration type allows to fully take advantage of cloud environment. At this level, the migration strategy should have been adopted, therefore the migration phase will focus on chicken little migration strategy which is suitable for migrating legacy applications within which data are moved first and then programs.

During the migration, all the frameworks focus in recovering knowledge from the legacy applications and then they implement the system from knowledge gathered. The transformation is based on modeling and transformation languages already defined in section 4.2.

4.3.2.1. Before modernization

Before modernization process starts, organizations should be aware of concerns that will have an impact on the modernization process, even if those concerns vary depending on use case, we will detail concerns that are more likely to be encountered :

1- Multi-tenancy. Multi-tenancy will depend on cloud service model chosen as well as on how the cloud provider has implemented multi-tenancy in the database. [P1], [P3] offer some tools to modernize legacy applications and apply multi-tenancy options desired.

2- Security. Security patterns in legacy application should be identified and checked in the new application to ensure that they are not broken [31]. It is also about taking services offered by the cloud provider in term of security. Security concerns will certainly have an impact on the framework chosen for authentication, authorization, etc.

3- Performance. The new application should have performance at least equal to the legacy application or even better. Therefore it is important to know which artifacts to measure and how to carry out measures.

4- Database. When a database modernization is involved, the release cycles will be longer and more complex and it may involve implementing wrappers or synchronization mechanisms in order to avoid functionalities overlapping.

4.3.2.2. Modernization

The transformation process of modernizing legacy applications goes in two phases : **Architecture-Driven Modernization** and **Model-Driven Architecture.**

1- Architecture-Driven Modernization (ADM). ADM is concerned about transforming the code, data and data schema into high-level Platform-Independent Models (PIM). For that the process goes through different phases where legacy application artifacts are transformed into PSM, thereafter the PSM is transformed into PIM, thanks to Reverse Engineering.

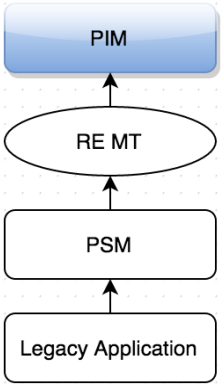

Figure 7: Reverse Engineering in modernization process

Per use case, database and softwares are transformed into conceptual models piece by piece during Reverse Engineering :

(a) Database Reverse Engineering (DBRE) [11]. Database Reverse engineering is about taking the legacy scripts, (DDL, physical schema, data), programs and transforming them into conceptual schema. According to [11], [8], transformation process has to go through DDL, script analysis, schema refinement, legacy logical schema extraction and conceptualization.

(b) Software Reverse Engineering. Software Reverse engineering is about extracting design information from the code source. For that the process has to go through static and dynamic analysis. The result of this process is conceptual models (e.g. UML).

2- Model-Driven Architecture (MDA). MDA is concerned about taking PIMs and PDMs and transforming them first into PSM with the use of Forward Engineering. Thereafter, PSMs are transformed in new application called modernized application that can leverage the new technologies and the cloud environment.

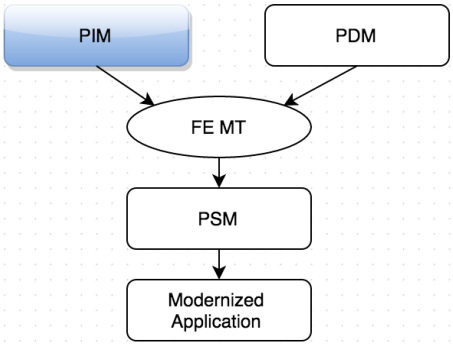

Figure 8: Forward Engineering in modernization process

Per use case, database and the software models are transformed into modernized application piece by piece during Forward Engineering :

(a) Database Forward Engineering (DBFE) [11]. Database Forward engineering takes conceptual schemas and transform them into new DDL and scripts. For that the process has to go through DB design, new logical schema extraction and the implementation[11].

(b) Software Forward Engineering. Application Forward engineering takes the conceptual models (e.g. UML) and transforms them into code source.

4.3.2.3. After modernization

Once the application has been modernized, it should be maintainable and testable. For that we should ensure to take care of these elements : firstly, the modernized and the legacy application should be functionally equivalent therefore there should be automated tests to prove that; secondly, the maintainability of the generated code should be evaluated by using suitable tools; Thirdly, the modernized code should also take into account non-functional requirements other than security, multi-tenancy and performance; Lastly, the modernized code should also follow organization standard and code quality.

Modernization process offers many benefits : Firstly, modernization allows to have the same business functionality on the legacy application and the newly created application. According to [6], modernization extends the useful life of an existing applications. Secondly, modernization promotes flexibility with the ability to derive from models, code that can be deployed on different

platforms including J2EE, .NET, etc. therefore resolving the interoperability problem and promotes the reuse of the domain models over time. Thirdly, modernization can automate the code generation therefore reducing migration and development cost. Besides the maintenance cost are likely to drop as organizations are maintaining models instead of the source code. Fourth, modernization allows the automation of the generated code allowing to have consistency and high quality code which is not prone to human error. Lastly, software development becomes again agile where organizations can adapt to changing circumstances as quickly as they wish.

4.3.3. Post-migration

Post-Migration phase is about deployment[P1][P2][P3][P4], validation[P1][P2][P3], testing[P1] [P2][P3], verification[P1][P2][P3], optimization[P1][P2][P3], maintenance[P1], withdraw[P3].

As we can see, [P5] and [P4] do not tackle the post-migration issues which is a disadvantage from their frameworks. Nevertheless, [P1], [P2] and [P3] seems to be the most complete study because they address the problems of optimization and leveraging the cloud environment after deployment.

	Artifacts	**Tools**
Deployment	Application, Data, Data schema	[P1], SaaS and Cloud provider tools
Leveraging the cloud	Security, Auto-scaling, Services, Dashboard, Multiple availability zones	SaaS and Cloud provider tools
Optimization	Database, Application, Tools, Monitoring, Usage	[P1], [P3], SaaS and Cloud provider tools

Table 11: Post-migration

Optimization is about about scalability of resources, infrastructures, software (distributed caching, etc.) and database (partition, clustering etc.). It is up to organizations to monitor and figure out which optimization is needed.

Post-migration phase is an important phase because it allows organizations to fully take advantage of the cloud environment and may save money according to the service or deployment models used.

4.3.4. Cross-cutting concerns

The cross-cutting concerns are concerns that cannot be cleanly decomposed from the rest of the system. They deal with all kinds of aspects : technical and general. They are important and the success of cloud migration will depend on taking them into account.

	Activities
[p1]	Business and organization aspects, migration Artifacts Reuse and evolution, target environment specification[40].
[p2]	Governance, Security analysis, Training, Effort estimation, organization change, distribution, multi-tenancy, elasticity analysis[12].
[p3]	Tools and new components to solve interoperability problems[39].

Table 12: Cross-cutting concerns in different frameworks

Even if some frameworks [P4], [P5] do not deal with cross-cutting concerns, organizations should be aware of activities that can be seen as cross-cutting concerns in order to deal with them accordingly. Of course, depending on the nature of the legacy applications some concerns may be important than others.

4.4. Cloud migration challenges

Legacy applications migration is a complex endeavor requiring re-analysis, source code reengineering, modernization, architectural changes, new strategies, cost and etc. However, the biggest challenge of cloud migration is how migration risk can be effectively identified and mitigated.

Migration is costly and time consuming. Even if this process is cheaper than rewriting, it still requires to mobilize resources. Large projects may require substantial investment which organizations should be aware. Nevertheless, as far as we are concerned, we believe in short term, the cloud migration process is costly but on the long term it can be beneficiary to the cloud consumer.

Skills and organization changes are inevitably in this case. Some roles may disappear, other may transition into other responsibilities therefore a change management support is necessary during the whole process. Besides, modernization introduces new technologies, new concepts and new tools therefore developers should be trained to use and understand the new application or the new environment effectively.

Business team believe that organization should put money and effort in things directly benefit the customer therefore their commitment in such endeavor may not be easy. The problem is

that the benefit and immediate ROI may not be visible as compared to the modernization costs being invested which may lead to them not getting the motivation in first place.

Coexistence of legacy and new application. The migration take time and sometimes the legacy application cannot be migrated at once. We may then see overlapping functionalities, data duplication, data synchronization mechanisms, temporary wrapper components during the interim phases with their own challenges as well.

Organizations should make sure to ensure business continuity. Business activity should not suffer from the migration process. New business requirement may still come on the legacy application and releases may still be planned. In the end not all the focus should be on cloud migration process.

Portability and interoperability risks are a real problem that can can lead to vendor lock-ins therefore undermine the cloud migration [45]. This risk can not only discourage organizations willing to adopt cloud environment but also it can prevent cloud consumer moving from one cloud provider to another one easily.

Service integration is another challenge. Even though, a successful migration is step forward, there is still another step to integrate the modernized application in the cloud with other applications on premises. During the migration process, some of the decencies were isolated but once the modernized application has been deployed, those dependencies need to be established one step at time.

Loss of control, organizations need to rely on third party and his tool in order to get more information on the migrated application. This implies a level of trust; the necessary tools are availability and they are providing pertinent information. This paradigm shift can be baffling in first place and organizations may not get them same satisfaction as when they had the whole control over their infrastructure.

Organizations should also understand how testing will be completed prior to and after migration. Therefore they should gather test data, testing environments should be available and tests should be automated. The challenges here is to prepare test cases prior to the migration and have test data that covers all the scenarios.

Organizations should be aware on how to ensure reliability through redundancy and failover. When migrating to the cloud, organizations should also understand the impacts on application performance and availability, and how these impacts will be measured. According to [45], organizations should be aware that metrics and standards for measuring should be established prior to moving in the cloud because the cloud provider may user different metrics.

Cloud consumers should also be aware of areas of critical focus for the security of the application running in the cloud and they should know how to resist, detect and recover from the threats or attacks. The challenge here is to be aware of those vulnerabilities.

Compliance with standards and governance can be challenging. Organizations should assess it correctly when the cloud provider is being chosen. They should be aware of legal risk created by the distributed nature of the cloud. They should be aware of which regulatory they are facing and the consequence of it.

4.5. Project management

The cloud migration of legacy applications is a project management process where a group of people work efficiently towards successful completion of a software project. Each member has to be aware of his responsibilities in order to achieve the group goal.

During the migration process, the elements below should be considered:

1- Use an iterative and incremental approach. The approach allows to get ROI; reduce risk and shorten time to market. Frequent planning allows also to reduce unpredictability. Moreover, customers are more involved and developers learn best practices from previous iterations and apply them in the next iterations.

2- Follow Software Development Life Cycle (SDLC). Even though the modernization effort follows an MDA approach, the project should follow the full SDLC of project planning, architecture, design, development, testing and production launch.

3- Prioritization and risk handling. Identify, analyze and create abatement procedure to mitigate the risk. Risky elements should be on high priority than other elements.

4- Estimate. Estimate is not enough, it should be a good estimate which can provide a clear view of tasks being carried out and lead to achieve the project objective. According to [43] underestimation creates numerous problems while overestimate allows Parkinson's Law to kick in. The estimate can also help in mitigating cost risk in case a task is taking more time than expected.

5- Measure. The progress should be measured, therefore key progress indicators (e.g. lines of code, artifacts, etc.) should be defined in earlier stage, so that they can be tracked. Tools (e.g. integration continue tools, etc.) can help in having a more clear view on measures.

4.6. Synthesis

In this chapter, we have analyzed five frameworks. This has highlighted that the pre-migration phase is important in order to make cloud migration a success. Although, there are differences in frameworks, they all agree in modernizing legacy applications in order to fully take advantage of the cloud environment. The modernization is an important phase that allows the new application to be transformed into cloud-enabled application. The post-migration is about deployment, leveraging cloud environment and optimization. The cross-cutting concerns should also not be neglected because they are concerns that affect the smooth running of the project.

On one hand, we should also not loose focus that cloud migration process is also about project management. Therefore we should know that incremental planing, prioritization, estimation and measurements are key to project success. On other hand, a good strategy is necessary in order to make a good planning that can lead to a successful migration.

Cloud migration challenges exist and knowing them can help organizations avoid optimistic assumptions. Challenges can help grasp the complexity of cloud migration process and help the team to have a good understanding of challenges ahead. Knowing the challenges will certainly have an impact on the planning as well.

However in order to mature the field further, we agree with [12] that cloud computing and software engineering researchers need to propose a common research agenda because there are similarities in their frameworks. Many researches have also emphasized the the lack of exhaustivity[12] when it comes to tools during the cloud migration process and afterwards, those tools face the same challenges as any open source tool which are lack of vendor support and unfamiliarity.

<table>
<tr><th></th><th colspan="2">Steps</th><th>Output</th></tr>
<tr><td>Pre-migration</td><td colspan="2">Feasibility and assessment studies (Business and Technical)</td><td>Migration plan</td></tr>
<tr><td>Migration</td><td colspan="2"><table><tr><th>Steps</th><th>Output</th></tr><tr><td>Reverse engineering</td><td>Models/reusable elements</td></tr><tr><td>Forward engineer</td><td>Loose coupling Application/DB</td></tr></table></td><td>Modernized Application/Database</td></tr>
<tr><td>Post-migration</td><td colspan="2"><table><tr><th>Steps</th><th>Output</th></tr><tr><td>Deployment</td><td>Testable Application/DB</td></tr><tr><td>Leveraging cloud</td><td>Economies of scale</td></tr><tr><td>Optimization</td><td>Scalable Application/DB</td></tr></table></td><td>Cloud-enabled Application/Database</td></tr>
</table>

Table 13: Migration synthesis

Chapter 5

5. Discussion

5.1. Cloud concerns

Many organizations have celebrated migrating their legacy applications to cloud but they are mainly moving simple applications leaving aside others. Organizations face many constraints whether they are economic, legal, technical, political, historical, governance, etc. Some may be easy to overcome but others may be more difficult. In any case, they have to be assessed and in order to have a good understanding of them and maybe find a solution. Organizations should know which cloud migration constraints they are facing and what it takes to resolve them and the risk involved.

Organizations concerns in adopting cloud are mainly :

Organizations would like to have security and privacy for their own data. Indeed, it is the most important asset they have and they would like to protect it. The challenges organization faced, is about sharing data with legitimate users while protecting personal information from unauthorized access. Another concern is whether a third party can have access to the data and can read them without user consent. This challenge is not easy to solve though because the problem exist in all service and deployment models even if the levels are different.

Loss of control may be also a challenge to overcome for organizations. The lack of visibility may hinder the relationship between the cloud provider and the cloud consumer. In fact, some service and deployment models, may introduce an abstraction layer that prevent the cloud consumer to have a direct access to some parts of the cloud which may leads to no clear definition of responsibility between both actors.

Another concern is the contract definition. The contract should protect the cloud consumer, in case of conflict (e.g. the cloud provider cannot commit to service agreed, he goes bankrupt, etc.), the cloud consumer should be able to leave the cloud provider easily. The contract should also contain measurements agreed upon (e.g. performance, availability, etc.)

Organizations do not know how to get started and they get lost in all those terminologies and paradigms. With legacy applications, even a small changes can have big repercussions on the business. So, coming and say that the legacy applications which is somehow a black box to the developers, need get transformed, is quite challenging and not an easy process to accept. However, organizations need to overcome that fear in order stay competitive on the market.

Organizations should welcome the migration to cloud so they can to be more agile and deliver faster.

5.2. Recommendations

We suggest these recommendations in order to motivate organizations to migrate their legacy applications, these recommendations should solve some of the problem raised in migration challenges :

1- Understand Service Level Agreements (SLAs). SLAs can be defined as *an agreement between a IT Service Provider and a Consumer* (ITIL v3). The SLAs should describe the service, the scope, the service quality and the responsibilities[44]. SLAs should state who are involved in agreement and should specify the responsibilities of both actors. It should describe the service and deployment model contracted, metrics to respect, what to monitor and the security capabilities of the cloud provider. SLAs should be realistic and should not only include availability requirement but also performance aspects.

The SLA should be negotiated if it does not meet the organization needs. An exit clause should also be part of the agreement in case either the consumer or provider wants to terminate the contract which can allow the cloud consumer to subscribe to another service or to migrate to another cloud provider or even more invest in its own private cloud. Cloud consumers should be allowed to negotiate SLA violation as well as SLA detection because most of the time it is implicitly their responsibility. The contract enforcement should include reward and penalty clauses for either exceeding or failing to meet them. SLAs should explicitly stated what is excluded[48] in the contract.

2- Understand cloud appropriate solutions. If the cloud consumer cannot fully trust a public cloud provider with its own application, he should look into these appropriate cloud solutions :

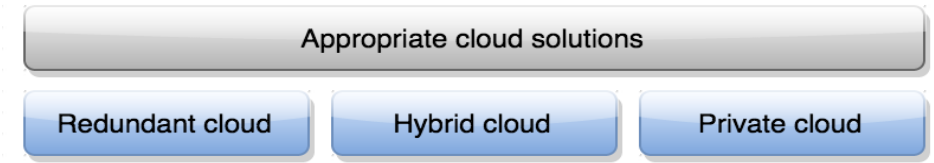

Figure 9: Appropriate cloud solutions

- Redundant cloud. It will allow organizations to ensure business continuity in case one of the cloud provider goes bankrupt or he cannot commit to his obligations. However, it may be expensive to adopt. Redundancy is possible on all the level of service models : IaaS, PaaS and SaaS.

+ IaaS. Redundancy is different whether it is a compute or storage service.

Compute Service	Storage Service
Redundant deployment	Fragmentation
Redundant computation	Erasure coding
Parallel computation	Replication

Table 14: IaaS Redundancy[42]

+ PaaS. Multiple PaaS but this can be problematic if the deployment is done on different cloud providers because of portability, interoperability, heterogeneity and geo-diversity issues.

+ SaaS. Multiple SaaS. According to [33], it may not be a good idea to use public SaaS for critical and High Availability (HA) applications because of unavoidable round trip delays of messages exchanged between SaaS consumers and cloud providers and nothing guaranties that the network will continue to provide acceptable service.

- Hybrid cloud. Hybrid cloud allows to leverage the existing investment. It can be an alternative solution by keeping sensitive elements on private cloud while other elements can be migrated in public cloud. This can address the problem of compliance by keeping control on some parts of the legacy application.

- Private cloud. Private cloud can be an option if organizations want to have the control of the whole system but private clouds do not necessary guarantee security. In fact, it is not because the application is running private cloud that it is secure.

3- Understand tools available. Tools allow to eliminate manual steps in migration process which are prone to human errors. Besides, tools help in time saving by automating some parts of the process. Tools assessment can be an important task during the pre-migration phase for legacy applications because it can show if the necessary tools are available as well as their compatibility in Pre-migration, Migration, Post-Migration and Maintenance.

Figure 10: Migration tools assessment

The figure above, shows some tools may come from a migration framework such ARTIST, other may come from the cloud provider. Of course others may be out there as open source or proprietary and not linked to any framework or a cloud provider.

Pre-migration	Migration	Post-migration	Maintenance
Business assessment	Reverse engineering	Testing	Monitoring
Technical assessment	Forward engineering	Deployment	Reporting
Cloud provider assessment		Migration verification et validation	Maintenance
		Optimization	Diagnostics
		Certification	

Table 15: Different types of tools

4- Make an incremental project planning. The release cycle should be kept short in production. If the cloud migration is well planned, it can improve both productivity and quality. The advantage of doing so is that faulty elements of the software can be quickly identified therefore it costs lower. It is also easier to implement and test an increment than the whole project. Besides, it can allow the team to focus more and conduct rigorous testing of each element. Customer are involved earlier and they can respond to features and review as needed.

5- Implement a Proof Of Concept (POC). The realization of a proof of concept can allow to demonstrate the feasibility of an idea. Compared to a prototype, a proof concept is cheaper. With a POC, developers can clearly see the tremendous potential of adopting cloud consequently motivate for its adoption. It can also allow developers to get familiar with new concepts, new technologies and new process. A proof of concept should stay small and may or may not be complete. However, the positive welcome of a POC, does not guarantee a successful migration.

5.3. Synthesis

In this chapter, we have presented cloud concerns. After that, we have drawn some recommendations in order to encourage organizations to migrate their applications into the cloud. These recommendations are important and constitute the first elements to be investigated. However the list is not exhaustive and organizations should keep updating it according to their specific requirements.

Cloud concerns	- Security and privacy protection - Loss of control - Contract - Get started
Recommendations	- Understand Service Level Agreements (SLAs) - Understand cloud appropriate solutions - Understand available tools - Make an incremental project planning - Implement a Proof Of Concept (POC)

Table 16: Cloud concerns and recommendations

Chapter 6

6. Conclusion

In this study of cloud migration of legacy applications, we started by defining the cloud computing, its actors, its essential characteristics, its deployment and service models, the differences between cloud and traditional environmental. Afterwards, we defined legacy system and we highlighted legacy system concerns with regards to scalability, system evolutions and migration strategies.

From those two preconditions, we analyzed five cloud migration frameworks : ARTIST, Cloud-RMM, REMICS, CloudMIG and Legacy-to-Cloud Migration Horseshoe framework. From that analysis, we found out that organizations are migrating simple applications (e.g. email) leaving aside legacy applications.

In the end, we decided to highlight cloud challenges and draw some recommendations that will encourage more organizations to move their legacy applications in the cloud. In the recommendations, we emphasized that organizations should firstly understand their agreement contract and more importantly it should contain an executable exit strategy in case one of the actors would like to cancel the agreement; secondly, organization should understand there are appropriate deployment models depending on use case, thirdly, they should assess tools that can help them during the cloud migration process. Fourth, organization should make an incremental project planning so that they can stay focused, lastly, to get started, organizations should implement a proof of concept.

Eventually, cloud experiences and adoptions is different from one organization to another. It is up to the organizations to figure out what their needs and aspirations are. Cloud migration of legacy applications can be a challenge but where there are challenges, there are also opportunities to do things better. As for cloud migration, it is an opportunity to archive, consolidate, or to decommission some parts of legacy applications allowing to focus more on domain problems. As for organizations, it is an opportunity to learn from the experience by applying incremental approach because the migration experience is never the same from one project to another.

On cloud providers' side, they need to instill confidence by publishing their internal processes, governances and compliance mechanisms. They should address the problem of interoperability and portability of their cloud services openly to avoid cloud vendor lock-in. And addressing those challenges will help more organizations move their legacy applications without many apprehensions.

On organizations side, they need to embrace changes which need to be carefully planned because it can bring employees resistance thereby undermining the whole process. Certainly following processes (e.g. ARTIST) is an important asset, but as it was shown in chapter 5, we need to highlight some important points depending on requirements of legacy applications. In fact, legacy applications need more focus on certain areas, those points should be assessed accordingly and deeper.

Cloud migration involves spending time and money therefore it should be motivated and it should involve all the stakeholders. It is a whole set of different ways of thinking with its own challenges and opportunities. After migrating a legacy application into the cloud organizations have to monitor and optimize their processes, they have to take ownership of their new environments or interfaces and use them wisely. Ultimately, cloud migration of legacy applications is not just about moving into cloud, it is also about transforming legacy applications, implementing the right strategy, planning, project management, communications, etc.

Future work can take this analysis a little deeper and draw recommendations based on different criticality of legacy applications. The objective being to help more organizations have confidence in themselves.

7. References

[1] Cloud Computing Patterns, http://www.cloudcomputingpatterns.org/, [version from 7 February 2015]

[2] Almonaies, Asil A., James R. Cordy, and Thomas R. Dean. "Legacy system evolution towards service-oriented architecture." *International Workshop on SOA Migration and Evolution*. 2010. http://ftp.qucis.queensu.ca/home/cordy/Papers/ACD_MigToSOA_SOAME10.pdf

[3] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges."*Journal of internet services and applications* 1.1 (2010): 7-18. http://u.cs.biu.ac.il/~ariel/download/ds590/resources/cloud/cloud_sota.pdf

[4] ARTIST EU Project, "Challenge - result", http://www.artist-project.eu/results

[5] IEEE cloud computing. "The premiere issue" http://www.computer.org/cms/Computer.org/ComputingNow/pdfs/CLC_20140501_May_2014.pdf, May 2014

[6] OMG. "Model-driven Architecture", http://www.omg.org/mda/

[7] Chris Birchall, Understanding the challenges of legacy projects, Re-Engineering Legacy Software, http://www.manning.com/birchall/

[8] Hainaut, Jean-Luc, et al. "Database reverse engineering." *Encyclopedia of Database Systems*. Springer US, 2009. 723-728. https://www.researchgate.net/profile/Vincent_Englebert/publication/226590879_Database_Reverse_Engineering_From_Requirements_to_CARE_Tools/links/0c96053abe06c6cf2e000000.pdf

[9] The Open Group . "Cloud Computing for Business" http://www.opengroup.org/cloud/cloud/cloud_for_business/what.htm, 2011

[10] Varia, Jinesh. "Best practices in architecting cloud applications in the AWS cloud." *Cloud Computing: Principles and Paradigms* (2011): 459-490. http://www.lifted-llc.com/docs/AWS_Cloud_Architecture_Best_Practices.pdf

[11] Henrard, Jean, Anthony Cleve, and Jean-Luc Hainaut. "Inverse wrappers for legacy information systems migration." *ISSN 0926-4515 All rights reserved editors: prof. dr. PME De Bra prof. dr. ir. JJ van Wijk* (2004): 30. https://pure.fundp.ac.be/portal/files/248204/200434.pdf#page=36

[12] Jamshidi, Pooyan, Ayaz Ahmad, and Claus Pahl. "Cloud migration research: a systematic review."*Cloud Computing, IEEE Transactions on* 1.2 (2013): 142-157. http://ulir.ul.ie/bitstream/handle/10344/3656/Jamshid_cloud.pdf?sequence=2

[13] Brodie, Michael L., and Michael Stonebraker.*Migrating legacy systems: gateways, interfaces & the incremental approach*. Morgan Kaufmann Publishers Inc., 1995. http://dl.acm.org/citation.cfm?id=SERIES9818.208444

[14] Wikipedia. "Software modernization", https://en.wikipedia.org/wiki/Software_modernization [version from 17 July 2015]

[15] ARTIST EU Project. "Open Source Package", http://www.artist-project.eu/open-source-package

[16] OMG. "Why do we need standards for the modernization of existing systems?", http://adm.omg.org/legacy/ADM_whitepaper.pdf

[17] Naono, Ken, et al., eds. *Software automatic tuning: from concepts to state-of-the-art results*. Springer Science & Business Media, 2010. https://books.google.be/books?hl=en&lr=&id=cmD5O83hS0QC&oi=fnd&pg=PR3&dq=%22software+automatic+%22&ots=jIFaDPWIT-&sig=eTlu1LkHbT1nbOmiSShgrlnYDZg&redir_esc=y#v=onepage&q=%22software%20automatic%20%22&f=false

[18] Wikipedia. "Cloud computing", http://en.wikipedia.org/wiki/Cloud_computing [version from 31 January 2015]

[19] Hogan, Michael, et al. "Nist cloud computing standards roadmap." *NIST Special Publication* 500-291 (2013). http://www.nist.gov/itl/cloud/upload/NIST_SP-500-291_Version-2_2013_June18_FINAL.pdf

[20] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011). *Special Publication 800-145* http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[21] Feathers, Michael. *Working effectively with legacy code*. Prentice Hall Professional, 2004.

[22] Bergmayr, Alexander, et al. "Migrating legacy software to the cloud with ARTIST." *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*. IEEE, 2013. http://www.researchgate.net/publication/261355920_Migrating_Legacy_Software_to_the_Cloud_with_ARTIST

[23] Bisbal, Jesús, et al. "Legacy information systems: Issues and directions." *IEEE software* 5 (1999): 103-111. http://csis.pace.edu/~marchese/CS775/Proj1/legacyinfosys_directions.pdf

[24] Mao, Ming, Jie Li, and Marty Humphrey. "Cloud auto-scaling with deadline and budget constraints." *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*. IEEE, 2010. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.307.8477&rep=rep1&type=pdf

[25] Amazon. "Migrating Web Applications to the AWS Cloud." *Migration Scenario* (2010). http://media.amazonwebservices.com/CloudMigration-scenario-wep-app.pdf

[26] Comella-Dorda, Santiago, et al. *A survey of legacy system modernization approaches*. No. CMU/SEI-2000-TN-003. Carnegie-Mellon univ pittsburgh pa Software engineering inst, 2000. http://www.sei.cmu.edu/reports/00tn003.pdf

[27] Kansal, Nidhi Jain, and Inderveer Chana. "Cloud load balancing techniques: a step towards green computing." *IJCSI International Journal of Computer Science Issues* 9.1 (2012): 238-246. http://core.ac.uk/download/pdf/25900769.pdf

[28] Alonso, J. Marcos, et al. "Cloud modernization assessment framework: Analyzing the impact of a potential migration to cloud." *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2013 IEEE 7th International Symposium on the*. IEEE, 2013. http://www.academia.edu/12375045/Cloud_modernization_assessment_framework_Analyzing_the_impact_of_a_potential_migration_to_Cloud

[29] Kopparapu, Chandra. *Load balancing servers, firewalls, and caches*. John Wiley & Sons, 2002. http://box.cs.istu.ru/public/docs/other/_New/Books/Administration/Load%20Balancing%20Servers,%20Firewalls%20and%20Caches%20(2002,%20Wiley).pdf

[30] OMG. "Architecture-Driven Modernization" http://adm.omg.org/legacy/

[31] Advanced software-based seRvice provisioning and migraTIon of legacy Software http://www.artist-project.eu/sites/default/files/MCF_Supporting%20document_0.pdf

[32] Open Group. "Setting Expectations and Working within Existing Structures the Dominate Themes for Day 3 of San Francisco Conference", http://blog.opengroup.org/tag/scalability/, February 2, 2012

[33] Synopsis, Cloud Computing. "Recommendations."*NIST Special Publication* (2012): 800-146. http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf

[34] Strauch, Steve, et al. "ESB MT: Enabling Multi-Tenancy in Enterprise Service Buses." *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.412.4511&rep=rep1&type=pdf

[35] Armbrust, Michael, et al. "M.: Above the clouds: a Berkeley view of cloud computing." (2009). http://www.moonther.com/cis492/abovetheclouds.pdf

[36] Ahmad, Aakash, and Muhammad Ali Babar. "A framework for architecture-driven migration of legacy systems to cloud-enabled software." *Proceedings of the WICSA 2014 Companion Volume*. ACM, 2014. https://www.researchgate.net/profile/Aakash_Ahmad/publication/262285532_A_framework_for_architecture-driven_migration_of_legacy_systems_to_cloud-enabled_software/links/54e619890cf2cd2e028c4c81.pdf

[37] Andrikopoulos, Vasilios, et al. "How to adapt applications for the Cloud environment." *Computing*95.6 (2013): 493-535. https://www.researchgate.net/profile/Vasilios_Andrikopoulos/publication/262889259_How_to_adapt_applications_for_the_Cloud_environment_Challenges_and_solutions_in_migrating_applications_to_the_Cloud/links/00b4953981929d1313000000.pdf

[38] Frey, Sören, and Wilhelm Hasselbring. "Model-Based Migration of Legacy Software Systems into the Cloud: The CloudMIG Approach." (2010). http://fg-sre.gi.de/fileadmin/gliederungen/fg-sre/wsr2010/28FreyHasselbing.pdf

[39] REMICS. "Introduction to the REMICS Methodology", http://methodology.remics.eu/

[40] ARTIST EU Project. "Vision", http://www.artist-project.eu/vision

[41] Müller, Hausi A., et al. "A reverse- engineering approach to subsystem structure identification."*Journal of Software Maintenance: Research and Practice* 5.4 (1993): 181-204. http://onlinelibrary.wiley.com/doi/10.1002/smr.4360050402/abstract

[42] Tobias Kurze, Markus Klems, David Bermbach, Alexander Lenk, Stefan Tai and Marcel Kunz. "Cloud Federation" http://aifb.kit.edu/images/0/02/Cloud_Federation.pdf

[43] McConnell, Steve. *Software estimation: demystifying the black art*. Microsoft press, 2006. https://books.google.com/books?hl=en&lr=&id=U5VCAwAAQBAJ&oi=fnd&pg=PT18&dq=Software+Estimation:+Demystifying+the+Black+Art+&ots=rVw8QzzoZ4&sig=JFHZJNNVZHHTZfnCznSCOu7poiI#v=onepage&q=Software%20Estimation%3A%20Demystifying%20the%20Black%20Art&f=false

[44] Wikipedia. "Service-level agreement", https://en.wikipedia.org/wiki/Service-level_agreement [version from 8 April 2015]

[45] Cloud Security Alliance (CSA). "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1", https://cloudsecurityalliance.org/csaguide.pdf

[46] Cattell, Rick. "Scalable SQL and NoSQL data stores." *ACM SIGMOD Record* 39.4 (2011): 12-27. http://www.sigmod.org/publications/sigmod-record/1012/pdfs/04.surveys.cattell.pdf

[47] Wikipedia. "Scalability", https://en.wikipedia.org/wiki/Scalability [version from 10 July 2015]

[48] Baset, Salman A. "Cloud SLAs: present and future."*ACM SIGOPS Operating Systems Review* 46.2 (2012): 57-66. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.295.9965&rep=rep1&type=pdf