MASTER IN COMPUTER SCIENCE

Self-evaluation tool for Software as a Service applications

Ihorimbere, Judicaël

Award date:
2015

Awarding institution:
University of Namur

UNIVERSITÉ DE NAMUR
Faculty of Computer Science
Academic Year 2014–2015

Self-evaluation tool for Software as a Service applications

Judicaël Ihorimbere

Internship mentor: Wim Codenie

Supervisor: _____ (Signed for Release Approval - Study Rules art. 40)
Philippe Thiran

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

Abstract

Software as a Service is one of the cloud computing service models. Cloud computing provides over the Internet configurable computing ressources to its users. Those computing ressources may be software applications in case of Software as a Service, platforms(programming environments) in case of Platform as Service and infrastructures(Physical and virtual machines) in case of Infrastructure as a Service. Software as a Service market just as well as cloud computing market in general is experiencing a strong growth. Therefore, Software as a Service providers need tools for self-assessement in order to measure their improvement, compare their applications with other applications and eventually improve their applications. Hence the need for tools of self-evaluation of Software as a Service applications.

In this thesis, an approach of self-evaluation of Software as a Service applications that is based on a design and implementation of a self-evaluation tool for Software as a Service applications for will be presented. More specifically we will design and implement a questionnaire tool that can be used by Software as a Service providers. The questionnaire will be based on a set of criteria defined from litterature review on Self-evaluation of SaaS applications.

Acknowledgments

In the preamble of this thesis, I would like to express my sincere thanks to all the people that provided their help and contributed in the development of this thesis and during my student's years at university.

First of all, I would like to express my sincere gratefulness to my thesis director, M. Philippe Thiran for his personnal support during my intership and the redaction of this thesis. He was always available during my internship and throughout the redaction of the thesis. He provided me much help for his valuable guidance, motivation and numerous corrections. This thesis would not have been possible without his precious help.

I would like to express my gratitude to my family for the support and the multiple contribution during the redaction of this thesis and during my student's years.

Then I would like to express my sincere thanks to the Software Engineering and ICT team of Sirris for the supervision during my internship. This thesis would not been possible without the interesting three-month internship that I have had the opportunity to perform at Sirris.

Finally, I would like to express my thanks to all my relatives and friends that supported and encouraged me throughout the realisation of this thesis.

Abbreviations

- **SaaS** : Software as a Service
- **PaaS** : Platform as a Service
- **IaaS** : Infrastructure as a Service
- **API** : Application programming interface
- **NIST** : National Institute of Standards and Technology
- **CPU** : Central processing unit
- **AWS** : Amazon Web Services
- **MSFT** : Microsoft Azure
- **OS** : Operating System
- **CRM** : Customer relationship management
- **MTBF** : Mean Time between failure
- **SLAs** : Service Level Agreements
- **DRP** : Disaster Recovery Plan

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

Software as a Service is one of models that compounds Cloud computing. This service model of the cloud computing offers softwares over the internet. The other main models are Infrastructure as a Service and Platform as a Service. They offer respectively infrastructures and platform over the internet. Today many companies are using the cloud computing. The market of cloud computing is trending for years. Among the 3 main models of cloud computing that are used, market of Software as a Service is the one that is more evolving comparing to the other service models. [10].

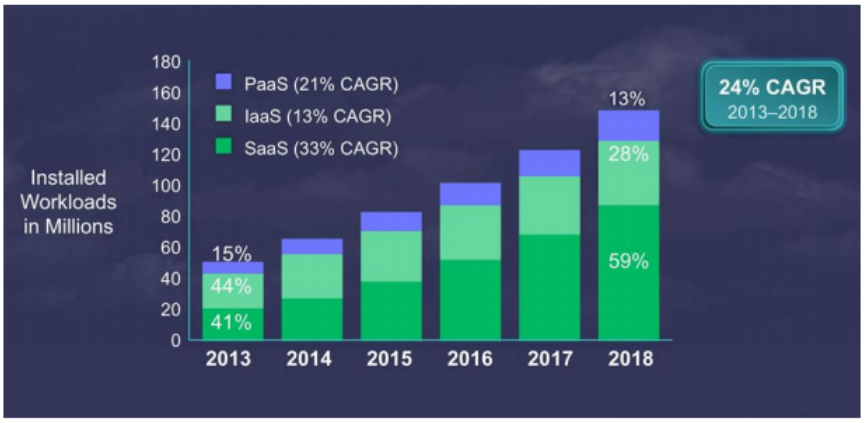


Figure 1.1: SaaS Most Highly Deployed Global Cloud Service by 2018 [10]

The rationale behind that trend may be understood when whe analyse shortly the benefits that are supplied by the use od Software as a Service applications. Software as a Service appears to be very interesting for their users especially for small and medium sized enterprises. Adoption of cloud computing solutions avoid them the heavy costs that would have be invested in investement and management of own computing ressources. The benefits are numerous : a low cost of entry, the avoidance of the responsibility of management of the software that is on the provider, a less risky investment, more safety for the data of the users, automatic back-up of data,

fast innovation of the Software as a service providers, more stability, model of offer much simpler than on-premises solutions, etc. [13].

The use of Software as a Service application does not come only with benefits. There are also risks. Among those many risks, we find the fact that the level of services promised by providers can be very different from the real level of services. There are also many risks linked to data security, business continuity in case of grave security problems, performance of the software, risks linked to the providers, or whether risks linked to legal aspects. Software as a service users have to balance the benefits and the risks of adopting such solutions. For the providers part,They sometimes do not have tools for demonstrating to the users if their applications meet the features that are described to be supplied. On the other side, they sometimes do not themselves have the insurance that the services they promise will be met. This is also due to the fact that they depend themselves to other providers. Providers of Software as a Service applications may sometimes need tools that can help them to assess their applications themselves.

The subject of this memoir is the self-evaluation of Software as a Service applications. As Cloud applications are very much used nowadays and givent the fact that users of Software as a Service applications are sometimes worried of the performance and the reliability of those applications, a self-evaluation of Software as a Service applications can be interesting for either providers or users. The self-evaluation of Software as a Service applications may be helpful for assessing the improvement of an application itself. Once a self-evaluation tool set up, it can be used at different moment. But firstly, the self-evaluation allow the providers to know where they stand given the objectives they have fixed when they perform their application. The self-evaluation tool can also help providers to compare themselves with other providers by comparing their applications given a serie of predefined criteria. As Software as a Service providers are plentiful and the Software as a Service applications numerous, a self-evaluation tool can help into establishing a set of characteritics that are common accross this plurality of applications. These characteristics, once set up, can be a basis for a certification of Software as a Service applications.

The second chapter of this memoir talks about Software as a Service and self-evaluation. Through this chapter, the key concepts about cloud computing are explained. The main characteristics of cloud computing are stood out in that chapter. The three models of cloud computing(Infrastructure as a Service, Platform as a Service and Software as a Service) are shortly described in general, before a deep definition of the Software as a Service model in the *section 2.1*. After defining the concept of Software as a Service, key actors of that cloud model are described in the *section 2.2*. The concept of self-evaluation of Software as a service applications is the focus of the *section 2.3*. Lastly, the last part of the chapter is the choice of the criteri on the *section 2.4*.

The third chapter of this memoir is concerned with the portability of the Software as a Service applications. Portability is described with a particular emphasis on the aspects that are relevant for the self-evaluation of Software as a service applications. Portability issues are covered on the *section 3.1*. Then the attempts of management of the portability issues are presented in the *section 3.2*. As the Software as a service portability cover different levels will be identified. The application portability will be the focus of the *section 3.3* and the data portability will be explained in the *section 3.4*. For these two last sections, on the one hand the two notions will described and on the other hand, an identification of what will be assessed will be carried out.

The fourth chapter talks about the the availability of the Software as a Service applications. After a definition and a description of the availability, different facets of availability are reviewed. The *section 4.1* presents the different availability level as providers supply their services with different level agreements. The *section 4.2* is concerned with redundancy of the Software as a Service applications. The *section 4.3* covers the backup of the applications and shows the different mechanisms that are used in order to insure availability. Monitoring of Software as a service applications is the subject presented on the *section 4.4* and the disaster recovery plan is the focus of the *section 4.5*.

The concept of interoperability of Software as a Service applications is addressed in the fifth chapter. After a description of the notion of interoperability, the *section 5.1* presents theoretically the approaches of addressing interoperability. The different interoperability scenarios are presented in the *section 5.2*. The *section 5.3* is a description of the syntactif interoperability that is a part of the components of the interoperability. The *section 5.4* talks about the other aspect of interoperability which is the semantic interoperability. The last part of that chapter , the *section 5.5* covers the the application programming interfaces, an important point about interoperability.

The sixth chapter of this memoir presents the Self-evaluation tool for Software as a Service applications. That chapter is explains firstly the methodology that guides the conception of the tool in the *section 6.1*. The *section 6.2* explains the question design and the method of scoring that is used for the questionnaire. The *section 6.2* is a short description and comparison of the tools that can be used for hosting the questionnaire. And last the *section 6.4* presents a typology and a serie of hypothesis that can be tested through the administration of the questionnaire

Chapter 2

Software as a Service and Self-evaluation

Software as a Service(SaaS) is one of the three service models of the cloud computing. The Cloud computing is a way of offering computing ressources over the Internet. It is an interesting manner for start-ups and small and middle companies that are starting an IT business. Indeed, using cloud computing helps to avoid onerous expenses. The cloud computing has been popularized by Amazon. Amazon has invested a lot in infrastructures(datacenters,...) for conducting its business. As its infrastructures were unused the great part of time, Amazon had the idea of renting its infrastructures to other companies that need computing ressources.

The concept of cloud computing has been used during the past ten years. The concept of cloud computing is sometimes misused. There are plenty of definitions of cloud computing available in the scientific research. Many researchers have tried to bring clarifications over the terms of cloud computing. Among the multiple definitions, we can retain this one : *" A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers."* [29]

There are many characteristics of the cloud computing. the National Institute of Standards and Technology(NIST) mentions 5 main characteristics that stand out [25]:

- *On-demand self service* : this is one of the basic characteristic of the coloud computing. The consumers can decide themselves to increase or decrease the resources supplied by cloud computing providers. This characteristic of cloud computing corresponds with the pay-as-you grow subsription. If the needs change within the time, the consumers is free to adjust his use of the ressource. The users are able to buy more or less to fit his need of ressources without the intervention of the provider. Cloud is linked with the utility computing. Com-

puting ressources are seen as public assets like electricity, water, natural gas, telephone network, etc.

- *Broad Network access* : ressources are accessed over the network and through many types of devices and platforms.
- *Ressource pooling* : ressources is shared by multiple consumers using a multi-tenant model.
- *Rapid elasticity* : ressources capabilities can be quickly increased at any time to meet the needs. Scalability is one of the main assets behind cloud computing. Users of cloud computing benefit of the fact that the ressources are very scalable. Given the fact that users can grow very quickly, the ressources and the software that are being runned in the cloud have to follow the increasing demand.
- *Measured service* : consumers pay only for the used capabilities and only for the time duration they used the service. Services are monitored and measured to allow quick and correct metering.

Cloud computing has mainly two deployment models [24] . There is the *public cloud* that is a deployment model that allows differents organizations to use the offered ressources. This deployment model is available in a pay-as-you-go manner to the public and constitutes this way *utility computing*. There is also the also the *private cloud* where the ressources are owned and managed in a private networks. Ressources are used internally by a single organization. This cloud deployment model is not opened to public.

At these two deployment models, the National Institute of Standards and Technology(NIST) adds two other deployment models of cloud computing [25] . On the one hand, the NIST identifies Community cloud that is a cloud computing that is shared by many organizations. Those organizations that form a community share the infrasture offered. . On the other hand there is *Hybrid cloud* which is a mix up of more than one cloud infrastructures(public cloud,private cloud, community).

Cloud computing is mainly delivered following three main models :

- *Infrastructure as a service* : providers supply infrastructure to consumers. The infrastructure are essentially supplied by data centres. The consumers can then rent servers, network, storage, processing where they can deploy their operating systems and their applications. This is made possible thanks the process of virtualisation. For each consumer, a virtual machine is created onto the data centre and the consumer can by this way deploy his operating system and application without worrying about the maintenance of the infrastructure, hardware that is in charge of the provider. Therefore, the consumer pay only for the used

ressources: essentially the number of virtual CPUs, the speed of CPUs and the random access memory available to the Virtual machine. [5]

- *Platform as a service* : providers supply platform and and an environment providing services and storage to consumers. This whole platform, environment and the storage is hosted in the cloud and is made vailable via the network . The consumers of cloud computing model are essentially developers that use the platform to develop, test and deploy their own cloud applications. Users are provided an environment where they can develop and run their own applications without managing the physical infrastructures and network under the platform. The PaaS platform is managed by PaaS providers that exploit infrastructures provided by IaaS providers. Users are mainly developers and profit the platform in a pay as you go billing.
- *Software as a Service* : providers supply application to consumers. Instead of buying applications and licenses that can be quickly very expensive, consumers rent service provided by SaaS providers in a on-demand pricing. By this way, users do not need install the software on their own computing infrastructure. The software is hosted on the cloud and is made available through the internet. The consumers don't need to manage the software, it is the task of the SaaS providers. The SaaS providers supplie the service and manage all related to maintenance and support. Via the chosen subscription, the consumers will profit of the service supplied via the SaaS. The consumer then pays only for the used capabilities.

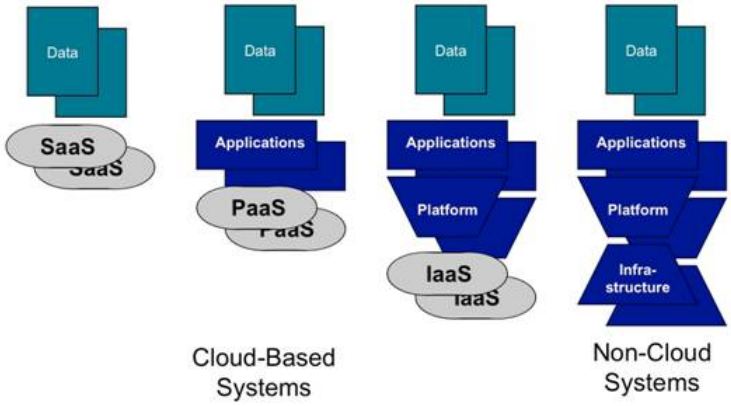


Figure 2.1: Cloud computing : models

2.1 Software as a Service : definition

Software as a Service has been defined by many authors. The definition we are going to retain is this following : *SaaS is defined as a model of software deployment via the*

Internet whereby the SaaS provider licenses an application to customers as a service based on usage or periodic subscription payments. SaaS software vendors typically host the application on their own Web servers or enable customers to download the application to consumer devices via the Internet [33]

Actually SaaS applications and mobile applications represents the most developped applications. Cloud computing offers ease of development of new applications. From the litterature, Tuomas Mäkilä et al. [32] analyzed the definition of SaaS applications. They reased 5 characteristics of SaaS that keep reverting :

1. SaaS applications are used through a Web browser
2. SaaS applications are not tailor made for each customer
3. SaaS applications do not include software that needs to be installed at the customer's location
4. SaaS applications do not require special integration and installation work
5. The pricing of the SaaS applications is based on actual usage of the software

SaaS Providers benefit from the opportunities and capabilities supplied by the Cloud ressources to develop quickly softwares that can be deployed and used quickly. Actually, SaaS applications are developed in a wide range of domains. There are a great number of SaaS applications that cover many business sectors: communication, human ressources management, company management, etc. SaaS is extending from day to day. Beside the major companies that provide SaaS(Salesforce, Oracle, etc), many small and medium-sized are developing their SaaS industry.

Three main actors of SaaS are identified: SaaS infrastructure providers, SaaS providers and SaaS consumers. [9] We are going to define them below.

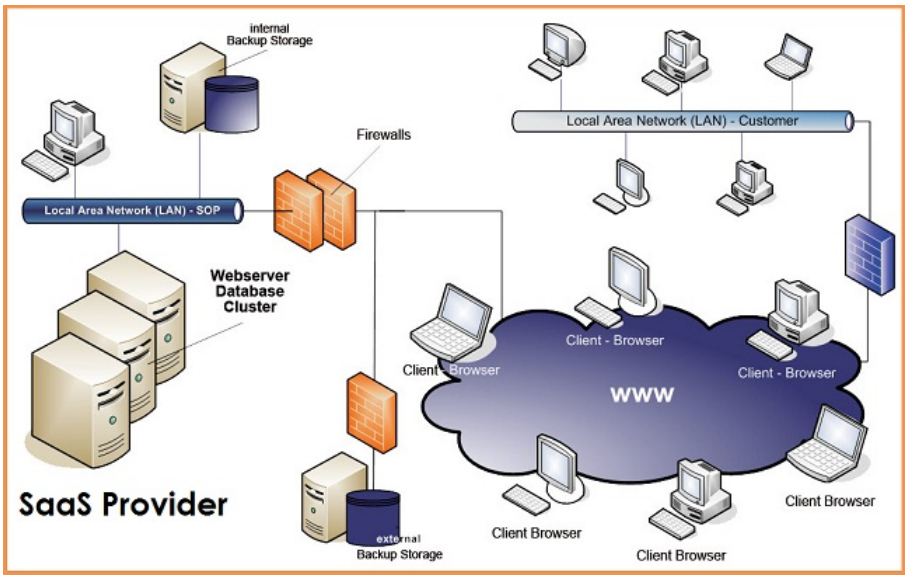


Figure 2.2: Software as a Service : overview

2.2 Software as a Service : actors

2.2.1 SaaS infrastructure providers

SaaS infrastructure providers are typically IaaS providers. Sometimes, they are PaaS providers. They are the owners of the computing ressources and they are responsible of their management. They manage hardware and software platforms required by their services. SaaS infrastructure providers are essentially great IaaS that have invested in computing ressources. For years, according Gartner studies, Amazon Web Services(AWS) and Microsoft Azure(MSFT) are the leaders among the SaaS infrastructure providers.

SaaS infrastructure providers bid essentially computing ressources. These providers usually have their own data centers. They exploit virtualization and provide virtual machine and network to their consumers(typically SaaS providers). The consumers will rent the ressources : there is no need for consumers to have their own servers and data center. they The consumers will just deploy their application on the virtual machines provided. Cloud infrastructures providers will invoice their consumer given the utilized ressources.

2.2.2 SaaS providers

The SaaS providers are generally PaaS and IaaS consumers. By using the advantages offered by the cloud computing, SaaS providers avoid by this way expensive costs

linked to ownership and management via on-premises solutions. They use the capabilities supplied by PaaS and IaaS providers to build their own applications and rent their applications. Once the applications developped they can be offered to SaaS consumers that can use them.

The provider will operate and manage the SaaS application. As SaaS applications are available over the internet, the SaaS providers may have to be sure that they can be accessed via many businesses and through many devices.

SaaS providers are responsible of management of the SaaS application. The offered services will be rent following a *pay-as-you-go* pricing model : pay per use. There are some SaaS providers that are well know because that are leading the SaaS market : Salesforce.com, Google Apps, Amazon EC2.

2.2.3 SaaS consumers

The SaaS consumers use the SaaS like any public utility: by this way consumers may only for what they have used, just as well we pay for electricity, water,... SaaS allows companies to save costs of initial investments. Delivered over the internet without installing anything. the service is accessed via a Web browser. It avoids to companies the expensive costs for operating and managing softwares.

SaaS consumers may choose among the packages offered by the SaaS providers the best one that suits to their needs. As SaaS are using the principle of *Self-service*, they are free to taylor the services offered. SaaS consumers are linked to SaaS providers via a contract : the service level agreements. This contract defines how the services are offered to SaaS consumers and shows clearly the responsibilities of SaaS providers.

2.3 Self-Evaluation of SaaS applications

Self-evaluation of SaaS applications can not be defined without defining well the concepts behind. In this section we will first explain what is an evaluation. Then we will explain the notion of self-evaluation. Lastly we will define what is Self-evaluation of SaaS applications.

2.3.1 Evaluation

Evaluation is a systematic and objective assessment of an object, an entity. Marvin C. Alkin defines the evaluation as *the activity of systematically collecting, analyzing and reporting information that can be used to change attitudes or to improve the operation of a project or a program* [6]. It is a process that is conducted in order to give a useful feedback that is used afterwards to make decisions. Evaluation is a process that is not

easy to achieve. It has to follow a strong methodolody in order to give a useful feedback. Evaluation will be useful to assess the value of something or that is ongoing or yet achieved. From an evaluation we can see if a set of objectives are achieved, if a certain level of performance is reached.

Typically evaluation process can have many kind of objects as targets . The objects of an evaluation is called evaluand. Evaluands can be clustered in 6 categories: Projects, programs, or organizations, Personnel or performance, Policies or strategies, Products or services, Processes or systems, Proposals, contract bids, or job applications[1]. Each of these evaluands can be assessed and there exists many methods that can be used. In our case, the evaluands are software as a service applications.

Evaluation process involves many related process as scoring, ranking and grading[1]. In effect, as evaluation to be a systematic assessment of an object, this also incorporates measurement, quantifification and caracterisation. It is a description of a situation, an entity, result, performance that provides a useful feedback. The result is usually expressed as a numeric score or a certain grading on a given range. Scoring is the process that consists in attribution of a numerical number given that a determined scale. One scoring achieved, ranking is possible between two or several evaluands and grading based on the score resultant can be performed.

The evaluation process can be done by internal evaluators(participatory evaluation) or external evaluators(expert-based evaluation). The internal evaluator will is someone who knows well the environment of the evaluand. Via the closeness of the internal evaluator, there is supposition that the evaluator has a better knowledge. But this could lead to a bias. Contrary to the internal evaluator, the external evaluator is someone who is supposed to be external to the environment of the evaluand. This exteriority gives to the evaluator more objectivity.

Evaluation processes can be qualitative or quantivative. This depends on the characterization of information collected during an evaluation process : wether they are tangible, observable and measurable or not. [15]. Quantitative evaluations gives a score while qualitative evaluations give an appreciation of the quality of the evaluand. Finally, evaluation processes are done via on the basis of precise criteria. As a matter of fact, the evaluand is assessed from a certain angle.

2.3.2 Self-evaluation

Self-evaluation is an evaluation process that is applied to oneself. This is a process of evaluation carried out by internal evaluator. Self-evaluation is a systematic assessment of oneself development, performance, attitudes, achievements, progress. A self evaluation process provides as an output, a snapshot(feedback) of the current state. Self-evaluation is a process that is used in many domains as in companies, or in do-

main of education. It is a process that is growing given its importance [19].

Selfvaluation can be a basis for a certification process. Certification is the action of attesting for an organization, person, object that some characteristics are met. The aim of a certification is to verify that the assessed entity meet a bunch of specified standards. It is often executed by an external entity or organization. Certification provides trust. Software certification is a process applied to softwares. The object of certification is software. The process can be based on different viewpoints: Quality, Process, usage ...

Self-evalution should be a continuous process. A self-evaluation tool for SaaS applications is a questionnaire that has as evaluation object SaaS applications. The evaluation target are SaaS providers. Self-evaluation of SaaS applications provides a Snapshot of the general state about SaaS applications.

There is Few/No research in self-assessment of SaaS applications. Indeed current researches emphasize on external viewpoints. SaaS evaluation may identify a general identification of SaaS applications issues. SaaS Self-evaluation provides a good snapshot of the actual situation. It can provide a good feedback of SaaS application to SaaS providers. Based on measurement, quantification and characterisation of the SaaS, we can identify weaknesses and strength of the SaaS providers. Classification and comparison between providers are made possible also. A SaaS providers can complete a self-comparison given previous snapshots.

The results of a self-evaluation can be a a basis for future certification. Given an establishment and and a verification of standards fixed, we can easily have an approach that could allow a certification process to take place.

2.4 Choice of criteria

The subject of my research is the self-evaluation of Software-as-a-Service applications. The idea behind the questionnaire tool is that the evaluation of the SaaS application is going to be done by SaaS providers themselves. This is bit different from what is currently done by the major studies about evaluation of SaaS applications. The target of the research is more emphasized on SaaS provider viewpoint while the big part of the current studies are interested on the SaaS consumer viewpoint. The self-evaluation will give a map of the current situation about the SaaS applications. The map will be given by SaaS providers themselves.

Research shows that SaaS users attend many things from the SaaS applications. They expect to have a good quality level from the SaaS applications they are paying for. They are worry about the offered service and wish to have the best from their

application. They are worry about many aspects : functionality, security, availability, network performance, resilience, service delivery management, organizational and financial stability, service level agreements, etc. These aspects represent risks they are exposed too when they are using SaaS applications. Reading research about SaaS shows quickly that some risks are recurrent.

Due to the fulness of the work, by eliminating and decreasing the area of the research, 3 criteria have been determined. The 3 criteria that have been retained are portability, availability and interoperability. These 3 criteria are one of the main concerns of SaaS providers. Indeed, SaaS providers are worried about those 3 criteria :

- portability : SaaS providers do not want to be confined within a unique SaaS infrastructure provider; then SaaS infrastructure providers should give them possibility to move their application and data utilized by the application another infrastructure provider. SaaS consumers may need to move their data to another SaaS provider
- availability : SaaS application need be available from anytime, anywhere and any devices within a minimal time response through the internet. This criterion is very important. It should be noted that a default of availability can provoke the bankruptcy of the SaaS providers
- interoperability : SaaS applications need to allow integration with other applications, but also exchange of data within other application. SaaS industry is based on the API economy where SaaS applica use services from existing modules from Web services, and other SaaS applications.

Chapter 3

Portability

Portability is the ability to use components or systems lying on multiple hardware and software environments. SaaS portability is the ability to move a service including data and application from one cloud provider to another or between public and private environments within minimal service disruption. It is ability to move any component of SaaS application accross different environments. Given the Cloud computing, portability is compound with 3 levels of portability : Data portability, application portability and platform portabilty. [18] The two first aspects are the interesting kind of portability that are appliable for the SaaS model. Portability of SaaS is the ability to move application, data and services from one cloud provider to another one or between public and private environment. Portability of SaaS is also defined as the ability to move any component of SaaS application across different environments. By accross different environnements, it means different datacenters, different types of cloud(public, private, hybrid, etc).

Portability of SaaS is an essential aspect that SaaS consumers are worried about. As there are many SaaS providers, consumers that wish to move from on-premise solutions have a difficulty to chose the best SaaS providers. SaaS consumers who are yet familiar with SaaS solutions would be tempted to move towards the best SaaS providers. However, SaaS applications are deployed in various environments. Each environment has its own specifications. Specifications can differ tremendously among the different SaaS. There is a lack of standardization between the different SaaS providers which may cause vendor lock-in. Nowadays, Vendor lock-in is one of the concerns of SaaS consumers.

From SaaS consumer viewpoint, portability is more related to data portability. SaaS consumers concern is the possibility to extract the data from a SaaS provider and load them to another SaaS provider. From SaaS provider viewpoint, service portability is their concern. This mean that the two levels of portability have to be achievable : data portability and application portability. Application portability is about moving the entire application or some of its components from one provider to another one. It is important for SaaS providers that data and application could be moved from one

Cloud provider(IaaS provider or SaaS provider) to another one.

Achieving portability across the Cloud is not easy. There are many issues that have to be tackled. In this chapter we will define the different what we understand by SaaS portabilility. We will cover the 2 main aspects of the portability : service portability, application portability and data portability. Then, We will see a short presentation of the issues related to SaaS portability. Finally we will present the main approaches that come to be the main solutions proposed to resolve this issues.

Service portability is the ability to move a service including data and application from one cloud provider to another. It may be understood by moving all the data and application that is deployed on a cloud provider to another one. But it can be also the move of data used by the SaaS accross equivalent SaaS applications. We may understand extracting data from one platform and load them to another provider. A SaaS application is could be ported and run on different cloud systems at an acceptable cost.

3.1 Portability issues

SaaS portability is a risky aspect of SaaS adoption . Vendor lock-in is a real concern of the adoption and the portability of SaaS application. There are risk of being blocked by the SaaS providers. There are many questions about usability of data once exported, functionality of application once moved to another cloud provider. As an example, when SaaS providers use proprietary technologies , it can be very difficult to extract the data in a usable form and load theim and application to another provider [24]. Data lock-in tends to affects the SaaS consumers when they want to migrate their data to another SaaS provider. But lock-in can also affect SaaS providers when they are trying to move their application and the data from a cloud provider(IaaS, PaaS) to another one.

The migration of SaaS application can bring many issues linked to compatibility due to heterogenous environments of SaaS providers. SaaS environments are very differents : Programming languages, communication protocols, OS of the platforms, etc. K.Bhavya et. al analyzed cloud service portabilty in order to allow secure migration[20]. The SaaS application has to be install and run on a new environment. Once this done, installed the new environment, the application will need to be updated. The process of moving entirely a SaaS application goes by many steps. Each step can be difficult to accomplish. Platform OS of the new environment need to be assessed in order to see if it will be compatible. If it is not compatible, some modifications will have to be done. This can be the same for the virtualization platform. The application itself will go through adjustements in order to fit to the new platform. There may be problem of compatibility linked to middleware and other dependen-

cies. Last but not least, the user state and profile will have to be managed. All of this, of course, will have to be done with minimal service disruption.

The specificity of the SaaS environments can be a serious issue. As a matter of fact, an environment can use many other services. There are some import and export of SaaS applications SaaS industry has been using strongly external services offered through APIs. One service can be indeed a combination of two or several services. For example, the Oracle CRM On Demand is composed by a bunch of modules [27]: sales, marketing, service, call center, analytics, mobile, etc. These modules are using specific services from different APIs. Then, if SaaS consumers need to move to Salesforce CRM, they have to be sure that all the services offered by the former are present. In the case of a SaaS provider that wants to move to another cloud provider, portability means moving the application and its data from a cloud provider to another. The target provider is able to offer the same services or equivalent sustitutes. More, applications sometimes need specific configuration files. These configuration files vary from one provider to another. It is not always easy to accommodate this configuration files to make them compatible with other environments.

There finally issues linked to the variety of data representations and data formats. Data are represented in many formats within great difference of semantics from one SaaS provider to another. While proceeding data migration, issues about data formats have to be resolved before and after the migration. Finally, added to this, SLAs between SaaS consumers and SaaS providers are sometimes not enough clear. Procedures of application and data migration are not well defined. The clauses about data portability are not well obvious.

3.2 Portability management

3.2.1 Intermediation

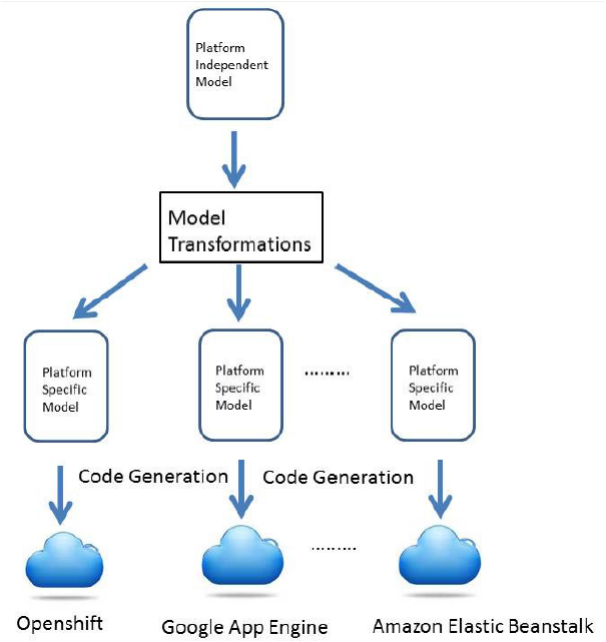


Figure 3.1: Intermediation : Model Driven Engineering in developing cloud portable applications[17]

The intermediation approach considers a different perspective. Instead of standardization of application and data, the application is decoupled from specific APIs and formats. An intermediate layer is introduced and this layer must be platform agnostic and can wrap and hide proprietary APIs. Through model transformation via Model Driven Engineering, this intermediate layer can the decoupled application to proprietary APIs. Utility of this intermediate layer is to map the different proprietary APIs through model transformation. The main difficulty is here the great heterogeneity of APIs. This is heavy to write model transformation to each API.

3.2.2 Standardization

The standardization approach consists on a definition of a common set of standards for application and data, and then the adoption of those standards by cloud providers[16] . A great difficulty is brought by this approach: there is no guaranty of acceptance of this standards by the major cloud providers. Each major cloud provider hopes for gaining more consumers on the market. Due to hard competition among the best providers, there is less possibility to agree about a standard that will be accepted by

all of the providers. Even a standard established, this is not sure that this standard will be adopted by those providers. Cost of adoption of the same standard can be indeed high.

Table 3.1: Standardization : Open-source libraries that support a certain degree of application portability[12]

Acronym	Link	Short description
$\delta - Cloud$	deltacloud.apache.org	It is REST-based API written in Ruby necessary to connect to various Cloud providers (Amazon EC2, Eucalytus, SmartCloud, GoGrid, OpenNebula, RackSpace, OpenStack and others)
Dasein Cloud API	dasein-cloud.sourceforge.net/	Java-based library for the access to compute services such as a virtual machine support, volume support, and other features associated with cloud compute as a service
fog	fog.io	Ruby-based library providing a high level interface to various Clouds 'collections' (images, servers). 'Requests' allow to dive in service particular features. 'Mocking' allows simulations by an in-memory resource representation.
jclouds	www.jclouds.org	It is an open source Java library that introduces abstractions aiming the portability of applications. It support more than thirty Cloud providers and software stacks including AWS, GoGrid, vCloud, OpenStack, Azure.
libcloud	libcloud.apache.org	Apache Libcloud is a standard Python library abstracting the differences between multiple Cloud provider APIs
Simple Cloud	www.simplecloud.org	It is a PHP library providing common interfaces for file and document storage services, queue services and infrastructure services

3.3 Application portability

To allow application portability, there are requirements that need to be met. On a SaaS provider viewpoint, three requirements are very important for the migration of the entire SaaS application [4] . This rely on the type of cloud provider that supplies the SaaS provider :

- If the Cloud provider is an IaaS provider, application portability for SaaS provider means ease of migration of Virtual machines(VM) and data from one IaaS provider to another. This means that the displaced virtual machines should be loaded to another IaaS provider at a minimal costs and changes.
- If the Cloud provider is an PaaS provider, this means that the application should be movable from one platform to another with eventually a minimal changes. These requirements need to be met to allow a SaaS provider pretend that the SaaS application is enough portable. The last requirement is the ability to move all the data from a cloud provider and load them fully on another cloud provider.

Application portability will depend on the specification of the environment but also on the requirements of the new environment. The requirements of the environment where the application is initially deployed must be the most similar to those of the environment where the application is going to be deployed. If the two environments are quite different, valuable effort must be done to allow the application portability. It is the same case when we are moving only some components of the application. SaaS portability incorporates also the ability to move components of application. An application which can be deployed on many providers, Either in private cloud, public cloud or hybrid cloud instead of only one shows a high level of portability.

The environment where the application is going to be moved must offer an interface that is enough compatible for the application which is moved. Application portability is the ability to move the entire application. This can mean to move a virtual machine or an instance of the application is deployed. This requires the specifications and the needs of the application have to be anough compatible with the requirements of the environment. The solutions are mainly some applications enablers like APIs that specify well how input and output are managed. A good description of management of input and output is important to move the entire application.

SaaS portability will strongly depend on the cloud provider(IaaS, PaaS) and the tools the provider uses. Cloud providers offer many tools and facilities to SaaS providers to allow them develop their applications. For example, in a case where the SaaS provider is a customer of a PaaS provider, the latter offers a platform the former with specific characteristics. The offered platform has its own configuration that may be whether or not opened. That will affect strongly the compatibility of the SaaS application with other platforms. For example a SaaS application developed on Forces.com

using the programming language Apex will be hardly movable on a cloud provider that supports application developped using .NET like MS Azure. The developping tools can be a barrier when the SaaS provider want to export the SaaS application.

Self-evaluation of SaaS application covers 3 aspects of application portability :

- **experience about application portability** : 4 dimensions will be evaluated. The self-evaluation will focus the move of the SaaS application from on-premise environment to cloud environment, the move of the SaaS application from one cloud environment to cloud environment, the move of the SaaS application from cloud environment to on-premise environment and finally the possibility to add, remove and configure components dynamically on the fly.
- **duration of application deployment from development environment to runtime environment** : Development environment and deployment environment may be different. A quick and easy or a heavy migration from development environment to runtime environment indicates the level of the application portability. The duration will be graded on a scale. Self-evaluation will focus on the ease of move of SaaS application development environment to runtime environment
- **Ease of migration of the application from the current PaaS/IaaS provider to an other one laying on a different Operating System** : OSProgramming languages need runtimes and libraries. To ease portability, developers need to use standardized runtimes and libraries. But there can be great differences between OS, runtimes, libraries. Self-evaluation will focus of the cost of moving the application : will the source code need to rewritten, ease of untime libraries import of missed or the source will just need to be recompiled.

3.4 Data portability

Data lock-in is a heavy issue that SaaS providers and SaaS consumers have to deal with. Data portability can be indeed very difficult to achieve. Customers are afrait to be blocked within one provider and to be in a situation they can not move from this one to another one. The usability of exported data is linked to it, and in many situation, the SLAs are so complex so that it is not possible to the customers to understand all the details in it. Given the data pointview, data are in high heterogenous formats and environment, so migration is very hard to achieve.

The data lock-in is associated to data storage[17]. This is due to the fact that cloud providers may support different types of database. Yet data migration from a SQL database to a NoSQL database, data migration from a NoSQL database to another different NoSQL database may be fastidious fastidious. This is due to incompatible data structures. Another issue is linked to the query languages when we are querying

data from the databases. When quering languages are different, it is more difficult to move the data from one database to another one. Last, the export/import formats can increase the difficulties related to data migration. To avoid these issues, data should be in formats that are well standardized .The specification of data formats has to be well documented. The specification should define very well the syntax and the semantics of data formats.

Data portability need that the data to be platform-independant. SaaS providers need to work on data formats to render them standardized. But if the providers who are the market key-players have to adopt those standards, by this way, the standards will be recognized[12]. Data migration requires a good specification of the data formats. The specification of the data should be in a standardized form, in readable form for a quick and effective migration. Process of data migration can be quickly hard in the case where minimal effort has been done to standardize data formats. When the data are well standardized, as an example, the data should be in a machine-readable form, even taugh human-readable form is good start.

Self-evaluation of SaaS application will cover mainly 3 aspects of data portability :

- **likelihood of data lock-in** : Cloud providers(IaaS and PaaS providers) provide to SaaS providers multiple ways to build their applications. Data lock-in depends on the level of openess of tools offered by the cloud provider. The level of openess can be evaluated by assessment of the tools that are given to SaaS providers in order to develop their SaaS applications. From less to more open tools, we can mention these possibilities : Cloud providers can provide proprietary tools for online development via a Web browser, using visual interfaces and design templates. Another possibility is custom functionality via native APIs offers. They can also offer a possibility of deployment and development of arbitrary source-code. Last, Cloud providers can offer support for standards widely adopted and used technologies. These capabilities can be combined.
- **standardization of the data formats** : data formats is one of the concerns of SaaS self-evaluation. Standardized data formats are helpful for an easy data migration from one cloud provider to other providers(IaaS,PaaS). When data formats are standardized, data migration can facilitated by external tools. The idea here is to detect if the SaaS providers have to modify the data formats before data migration in order to make data suitable with the new cloud provider. We need to indetifiy if the data formats will be less, partially or fully modified. This has incidences on automation of the process. The automation of the process can then be either possible or not. If it is possible, self-evaluation will try to detect if the process is semi-automatic or fully automatic. By this way, SaaS providers can state how their data are portable or not.

- **Ease of data migration between different database models** : The SaaS application can compatible with different database models(relational databases,NoSQL databases). As there are huge differences between relational databases and NoSQL databases and also among NoSQL databases, the issue is to identify if the SaaS application is compatible within many database models. Ideally, compatibility with many different database models would be an asset. Unfortunately this have a heavy cost. Self-evaluation of SaaS applications will take into consideration this aspect. On the one hand, the process of self-evaluation will assess the ease of data migration that considering databases of the same model, and on the other hand, the process will access the data migration where the database models are different(either from relational databases to NoSQL databases and vice-versa or from NoSQL database to another different NoSQL database).

Chapter 4

Availability

Availability is the probability that a system, at a point in time, will be operational and able to deliver the requested services [2]. Availability is the proportion of time during which a system or a service is functional to the total time it is required. It is the probability that a system is functional. Availality is defined by the following ratio :

$$Ratio = [(Total responsetime–downtime)/Total servicetime] * 100 .$$

The ratio takes into consideration periods of reponse time, downtime and service time. Service time is the time which is defined in the SLAs. SaaS providers agree to guaranty availability of their SaaS application during that time. Downtime is the time during which the SaaS application is not available. The ratio will be balanced as a percentage. For exemple, if the availability of the SaaS application is 0.95 it means that the application is available 95% of the time

As SaaS to be delivered over the internet, ensuring availability of SaaS applications involves availabilty at many levels. The network, the system, the application and the application need to be available at any moment they are required. The causes of downtime are multiples. Causes of downtime can be from the network, the system, the data or the service itself. As a matter of fact, if there are failure due to the network, a failure due to the platform, a database failure or the service(application) failure, this will impact on the whole availability. SaaS availability depends on system robustness and on the avaibility of infrastructures service providers

Availability of SaaS applications contributes to service reliability. SaaS providers pay attention to eventual SaaS application failure and the duration of the failure. In effect, when failure occurs and durates over a significant period, this is risky for the sustainability of the SaaS business. Contrary to on-premises solution, availability of SaaS applications depends strongly of the capability offered by high availability architecture. Meanwhile, there are many precautions that need to be taken by SaaS providers in order to ensure a good level of availability[21]. For this purpose, it is necessary to implement fault avoidance, fault detection and removal and fault-tolerance

mechanisms. Mechanisms for avoiding single points of failure, detection of failures. These mechanisms are also helpful for insuring security of the SaaS applications and the data. SaaS providers need to do local storage, or a local backup of the mission-critical data. This can help in case the cloud infrastructure provider is unable to deliver the service as planned.

4.1 Availability level

Self evaluation of SaaS application takes into consideration the availability level supposed to be insured by the SaaS provider. SaaS Availability level is specified into SLAs. Many SaaS providers admits that their SaaS application are high available. The level of availability depends strongly on the availability of all the providers. There are SaaS applications that need high availabilty more than others. That's why SLAs define hours during which service is normally available or not. SaaS providers do effort to specify working hours(Business hours) from non working hours in their SLAs.

Consumers of SaaS applications expect to access the service as far as possible. SaaS providers are then pushed to improve the availability of their services to meet the telco grade standard of 99.999%[7]. However, in order to meet this standard, all the layers have to be capable to provide such level of availability. The cumulative Mean Time Between failure(MTBF) of the hardware components should be quite small. This will allow for the SaaS providers to meet the desired availability level. Beside the hardware components, network hass need to be as fast as possible.

Table 4.1: Availability level[7]

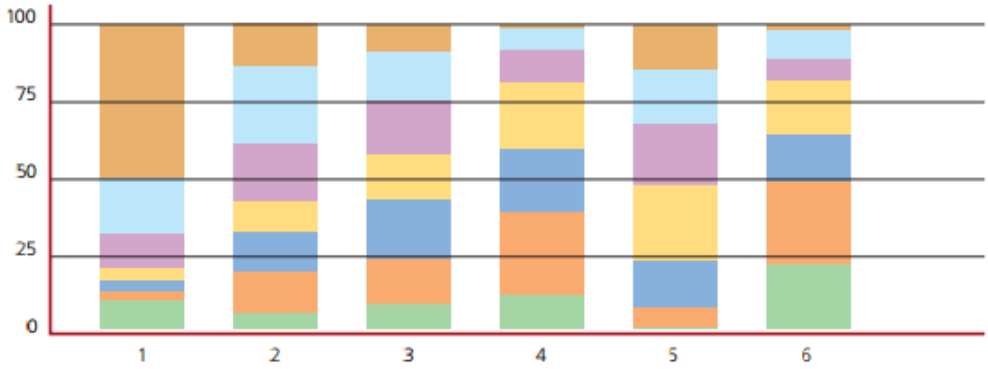
Availabilty(%)	Downtime Per Week	Downtime Per Month	Downtime Per Year
95%	8.4 hours	36 hours	18.25 days
99%	1.68 hours	7.2 hours	3.65 days
99.9%	10.1 mins	43.2 mins	8.76 hours
99.99%	1.01 mins	4.32 mins	52.6 min
99.999%	6.05 secs	25.9 sec	5.26 secs

Despite this, tolerance for downtime is different depending on the type of SaaS application. A survey of Rackspace shows that companies that use SaaS applications are differently sensitive for downtime. SaaS users were asked to rank their tolerance for downtime For example, 50.38% of the interviewees said they are least tolerant for downtime of Email and business productivity SaaS applications while only 9.42% were ranked to be least tolerant for Human ressources applications[28]. This means for SaaS providers, that availability level and tolerance for downtime vary with the

type of SaaS application.

Table 4.2: Tolerance for downtime[28]

	MOST TOLERANT 7	6	5	4	3	2	LEAST TOLERANT 1
1. Email and business productivity applications (e.g. spreadsheets, document creation)	9.92%	2.29%	3.82%	4.58%	10.69%	18.32%	**50.38%**
2. Customer Relationship Management (CRM)	5.6%	12.8%	13.6%	10.4%	19.2%	**24.9%**	13.6%
3 Enterprise Resource Planning (ERP)/Supply Chain Management (SCM)	8.57%	14.29%	**20%**	14.29%	19.05%	15.24%	8.57%
4. Product Lifecycle Management (PLM)	11.76%	**26.47%**	21.57%	21.57%	10.78%	6.86%	.98%
5. Accounting/Finance	.86%	6.03%	16.38%	**24.14%**	19.83%	18.1%	14.66%
6. HR	21.37%	**26.5%**	16.24%	17.95%	6.84%	10.26%	.85%



Self-evaluation of SaaS applications analyses of the level of availability considering what is defined into SLAs. As levels of availability are different given the kind of SaaS applications, different levels of availability are required. There are some applications than need more availability than others. SaaS application are then classified in different groups given their levels of availability. SaaS providers will classify themselves their SaaS applications into categories of availability. The selected levels of availability are : Less than 90%, between 90% and 95%, between 95% and 97%, between 97% and 99%, more than 99%.

4.2 Redundancy

Self evaluation of a SaaS application involves evaluation of the mechanisms of redundancy linked to the hardware and software components. As a matter of fact, SaaS providers are responsible for delivering a reliable application as it is indicated in the

SLAs. A default of availability can be caused by a mismanagement of the redundancy of software and hardware application. That is clear that the hardware is managed by infrastructure providers, but in the eyes of the SaaS consumers, the whole responsibility of redundancy management is the responsibility of the SaaS providers. That is why, a good approach of management of redundancy should be taken into consideration.

Redundancy is defined at two levels :

- **software redundancy** : Major components of the software can be redundant to ensure a high level of availability. During, times where requests are numerous, redundancy of key components may help to support a pick of requests. A redundant copy of application is useful. In effect, in case of failure of the many instance of application, the redundant copy may be used. This is an one of the mechanisms of faut tolerance. That is why it is important to make redundant whole instances of the application present on the virtual machine. In effect if one instance fails other instances could take over.
- **infrastructure redundancy** : infrastructures redundancy will all the mechanism of redundancy set up to avoid failure. It is the redundancy at different infrastructures level : redundancy of power supplies, network connections, back-up storage, batteries, generators, cooling system. This is the responsibility of the infrastructure providers to guaranty the redundancy of all these elements. Even the entire VM in which the application is hosted can be redundant, just in case the main virtual machine crashes.

As the infrastructure redundancy supposed to be guaranteed by cloud infrastructure providers, Self-evaluation of SaaS applications will focus on the software redundancy. SaaS providers will assess the redundancy of their application components. Redundancy of the SaaS application will grade given the number of software components that are redundant. Ideally, all the components of the SaaS application should be redundant. But given the cost of redundancy, SaaS providers can adopt different options. The worst case is when the components of the SaaS application are not redundant. But this is very risky for a SaaS provider. SaaS providers will rather chose among these others options : minimal components of the SaaS application are redundant, only the crucial components are redundant, Most of the components of the application are redundant or all the components of my application are redundant. The last option is preferable.

4.3 Backup

SaaS providers must ensure data backup in order to set up a quick recovery just in case of devastation. This is a a good way for guaranteeing availablity of the SaaS ap-

plication. However data need to be well encrypted for safety reasons. As a matter of fact, data must be protected from leakage and unauthorized access[8]. SaaS backup is the responsibility of SaaS providers even the data are stored in cloud infractrutes supplied by Cloud infrastructure providers. As a precaution, SaaS providers could have local backups for key data just in case their infrastructure providers are out of service. In case of downtimes from infrastructure providers, they could then continue to supply their service to the SaaS consumers.

Self-evaluation of SaaS application covers 3 main dimensions of backup.

- **backup model**: backup can be done following 3 models: Hot standby, warm standby and cold standby. Each backup model offers a different level of availability. In Hot standby model, standby servers are available every to run the application in case of disaster. Replication is synchronous so that the data are directly available on hot back up site. The secundary site can provide availability within seconds or minutes. In Cold standby model, data replication is done on a periodically(asynchronous). Recovery may take hours or days and may require hardware, operating system and application installation for restarting the whole service. Warm standby model is a tradeoff between a hot standby and cold standby. Replication is at the same time synchronous and asynchronous, and recovery may take a little more time comparing to hot standby[14].A SaaS application that uses hot standby model will be better assed that the one using warm standby model. The latter will be better assessed than another that use cold standy model.

- **geographical dispersion of backup sites**: It is necessary to separate geographically the backup sites. In case of natural disaster(fires, floods, earthquakes, hurricanes,...),, having geographically separated backup sites protects against outages. Even though a great geographical dispersion increases cost of data duplication, it is better for protecting data availability in case of a disaster. Then SaaS applications would be better assessed in this ascending order: backup sites located at the same place - backup sites located in the same region - backup sites located in different regions - backup sites located in different countries - backup sites are located in different continents

- **backup frequency**: Backup frequency will strongly depends on the backup model. Critical applications will need continous backup. Then, backup frequency indicates level of availability of SaaS applications. SaaS providers will evaluate their backup frequency of their SaaS applications on a scale from months to seconds. SaaS applications with a short time between backups will be better assessed.

4.4 Monitoring

Monitoring is the ability to examine the health of the whole sytem. It is helpful in for quick detection of issues about the SaaS application and their addressing. Many things can be monitored : software, Service, ... SaaS providers are worry about the availity of their services. In order to ensure compliance with their SLAs they need to test the offers of their infrastructure providers. In order to achieve their level of availability, infrastructure providers need to give tools for monitoring their infrastructure. Many things can be monitored : software components, load balancers, power systems, network systems, etc

Monitoring is useful to SaaS clients and providers. So that they can verify if the service is their SLAs. Level of availability must meet what is specified in the SLAs. This issue is only feasible if there are mechanisms of monitoring that are set up. SaaS providers monitors their system, by monitoring also the system of their providers. This is one of the detection of faults or failure. The capability of monitoring is also given to SaaS consumers so that they can assess if the SLAs are conform within the level of availability. Monitoring involves oversight of the SaaS system itself but also oversight of cloud providers. The two levels must be monitored by the SaaS provider in order to find quickly solution in case of any matter.

Self-evaluation of SaaS application will be interested in evaluating the monitoring system. The evaluation will be checking of the presence or not of fault end failure detection mechanisms. Those mechanisms should be implemented at at different levels: application level, virtual machines level and infrastructure level. Another aspect that will be assed will be the fact the SaaS providers offer monitoring capabilities to SaaS consumers.

4.5 Disaster recovery plan

Disaster recovery plan(DRP) is an interesting aspect for Self-evaluation of SaaS application. The assessment of the DRP will consist in evaluating or not if it has been yet tested or not. It will focus on the experience of SaaS provider in testing failover and failback mechanisms related to the SaaS application. Failover is defined as the swith *when active processing of incoming transactions is switched from the failed primary to the backup site* while failback is defined as the switch *when the causes of the primary failure have been addressed and the switch is made back to the primary* [26].

SaaS providers will themselves assess the experience they have about the testing of their DRP. Just in case the SaaS providers have been yet exposed to disaster, they will assess the process of addressing the problem. This means from SaaS providers an evaluation of failback and failover mechanisms. SaaS providers will assess that

their recovery time is compliant with SLAs. If recovery time is small, then the SaaS application will be better better assessed following this scale : Recovery time is guaranteed in more than 24hours - Recovery time guaranteed between 12 and 24hours - Recovery time is guaranteed between 1 and 12hours - Recovery time is estimated in less than 1 hour.

Chapter 5

Interoperability

According to the ISO/IEC 2382-01 norm, interoperability is defined as "*the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units*". This definition, although generic, provides a strong basis for defining SaaS interoperability. The functional units represent SaaS applications. SaaS applications in effect communicates within other systems. Those other systems can be SaaS appplications or whether on premise systems. SaaS applications exchange constantly information and data with other systems and run disparate modules from other systems. The important progress in the domain of cloud computing and a better coordination of Web services allow SaaS systems interoperability.

As cloud computing to be a specific way of computing, cloud interoperability is more precisely defined as *the ability of cloud services to be able to work together with both different cloud services and providers, and other applications or platforms that are not cloud dependant*[23]. Through this definition, it is possible to bring to light the characteristics of SaaS interoperability. SaaS interoperabilility involves the ability of SaaS applications from differents SaaS providers to work together. SaaS interoperability involves also the ability of communication betwwen SaaS applications and differents platforms and infrastructures including on premise platfomrs. SaaS Interoperability means a collaboration between SaaS applications that run on different environments.

Rashidi et. al. defined Cloud interoperatibility as *the ability of interaction of a service with other homogeneous or heterogeneous services to improve its service which may be implemented under one domain or different domains*[30]. On the one hand, homogeneous services mean that here interoperability is between the same cloud computing service model. As the concern is about SaaS interoperability, homogeneous services will appoint interoperability between two SaaS applications. On the other hand, heterogeneous services mean interoperability between different cloud computing service models. Here, it will be interoperability between SaaS applications and either PaaS or IaaS.

Based on the cloud interoperability definition, SaaS interoperability will be the ability of interaction of SaaS applications with other SaaS applications or other cloud service models(PaaS and IaaS) providers. The aim of this interaction is mainly to improve the services offered by the SaaS applications. Interoperability of SaaS applications involves their interconnectabity with other systems, their interaction and communication with others SaaS applications. SaaS interoperability can involve also integration into SaaS application, services from other SaaS applications and vice-versa.

Technically, in order to achieve SaaS interopabilility, many prerequisites have to be fulfilled. These prerequisites involve mechanisms that insure compatibility of different processes, used protocols and data exchange formats. Achieving SaaS interoperabilitity meets many issues. There are issues related to syntax(naming conflicts, data representation conflicts,etc) referred to syntactic interoperability on the one hand, and on the other hand there are issues related to semantics(data semantics, Logic and process semantics, etc) referred to semantic interoperability.

5.1 Interoperability management

There are many approaches that are used in order to manage interoperability. Those approaches are most of time theoretical. A significant concern about those approaches is the fast increase of SaaS applications. This renders the task of interoperability management complicated. SaaS providers can explore many interoperability models / frameworks as orchestration layer approach and adapters approach. These approaches once set up, are key sets not only in interoperabity management but also are advantageous in portability management of SaaS applications. SaaS providers may be encouraged examine the different approaches.

Magdalena Kostoska et al. [23] compare different approaches for management of Cloud computing interoperability. Among the approaches ascertained, two may suit the SaaS model :

- **orchestration layer**: This approach consists in insertion of platform layers where SaaS providers can register their services. It would be then possible to invoke directly the services offered by the SaaS applications by using the orchestration layer. This approach may be applicable to the other models(IaaS and PaaS).
- **adapters** : This approach consists in frameworks that provide interoperability between incompatible SaaS providers. Those frameworks may be set up either-for SaaS applications that are generally used or for specific SaaS applications. This approach is application for the SaaS model.

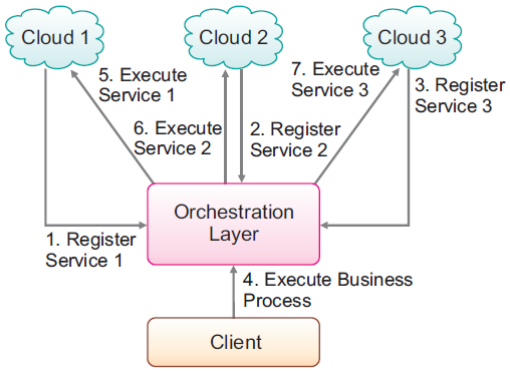


Figure 5.1: Cloud Orchestration [3]

5.2 Interoperability scenarios

SaaS Interoperability covers scenarios or categories [23] :

- **Interoperability of SaaS applications inside a single cloud** : SaaS applications exchange data inside the same cloud. The environment is homegeneous.
- **Interoperability of SaaS systems across different cloud environnements** : SaaS applications exchange information across different cloud environments. The exchanged pieces of information cause operations from one cloud to another to start. The environments are heterogeneous.
- **Use of SaaS application through a unified manner** : SaaS applications from different environments interoperate together and are accessed via a unique management system. The environments are heterogenous, and even the SaaS applications are different, they are managed through a unified manner.
- **Migration of a cloud application from one environment to another one** : SaaS applications and their data are moved from one cloud environment to another one. This category is in fact currently called SaaS portability. This has been developed in the chapter about portability

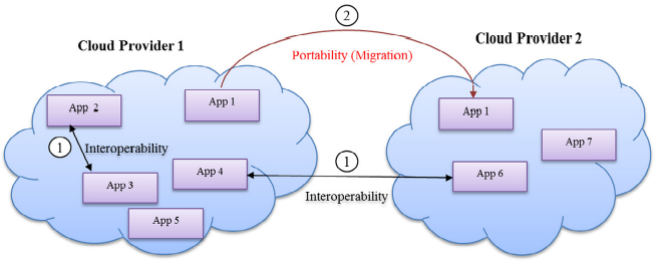


Figure 5.2: Interoperability and portability [22]

Self evaluation of SaaS application explores the different scenarios of SaaS interoperability. As SaaS interoperability means that SaaS applications can interoperate with other applications from different environments, Self evaluation surveys how many scenarios are covered. SaaS providers have the interest in the fact that the applications cover up the major part of the 4 scenarios. A SaaS application that interoperates or homogenous environments either in heterogenous clouds is more interoperable than an other one that can only interoperate inside the single cloud. Then if the SaaS application is able to interoperate with on-premise systems, with other SaaS application inside the same cloud, with other SaaS applications in heterogenous cloud providers, this will be advantageous for the SaaS interoperability.

When adressing SaaS interoperability, SaaS providers have to deal with standardization of data exchange formats. The ultimate aim is achievement of dynamic interoperability with random SaaS provider. Data formats have to be standardized in order to perform easily a dynamic exchange of data between different SaaS providers. Self evaluation of SaaS applications will assess how easy is exchange of data by assessing the eventual difficulties to achieve interoperabily. SaaS providers assesses the ease of interoperability in the case of achieving interoperabilityin. The following levels represent the degree of difficulty: Data formats have to be fully modified - Data formats have to be partially modified - Semi automatic data exchange is possible - fully automatic data exchange is possible

5.3 Syntactic interoperability

Syntactic interoperability is the the ability to exchange data. In practice, this is limited on the 3 following points : First the description of what a SaaS application is able to do, then the description of where the SaaS application resides, and finally a description of how to invoke the SaaS application [31]. Pushing syntactic interoperability far will include service discovery issues, service description , management of heterogeneities at message level, management of discovery and invocation of the SaaS application at runtime, management of dynamic service discovery and dynamic

service invocation.

There are many issues to resolve in order to achieve interoperability of SaaS applications. Those syntactic issues are:

- **Naming conflicts**: SaaS providers and differents cloud providers do not have the same level of naming. A concept can be defined differently given the SaaS application. The great number of SaaS providers do not ease the uniformity of naming.
- **Data representation conflicts** : SaaS providers define their data in accordance to their needs. For each and many SaaS providers, even they are in the same business domain, have differents data representations.
- **Standardization** : Use of different standards may increase the level syntactic interoperability. When services are defined using SOAP, it is not easy to SaaS providers that use WS-* or EbXML to easily discover and invoke them

Self-evaluation of SaaS applications involves that SaaS providers assess if the different levels of syntactic interoperability are covered. This is the first step of achieving SaaS interoperability. SaaS providers will the assess if their SaaS application can easily exchange information with other SaaS applications. Use of many different interoperability protocols and programming languages by SaaS providers implies that SaaS providers specify very well their SaaS applications and data using different syntaxes. This is characterestic of a good management of SaaS applications. SaaS providers will analyze also if their SaaS applications offer an application programming interface for an ease of integration of their SaaS applications in different systems.

5.4 Semantic interoperability

Semantic interoperability is the ability to operate on the exhanged data. Semantic interoperability involves the description of what a SaaS application can do, the description of where the SaaS application resides, the description of and how to invoke the SaaS application with meaning and interpretation of the data and operations at runtime [31]. Beside issues linked to syntax, effort about semantics need to be done. Nagarajan et al., 2006 identify 4 types of semantics for any application description. Those types are system semantics(deployment, Load balancin), data semantics(Typing, Storage, manipulation restrictions), non-functional semantics (performance, security) and logic and process semantics(Programming language, Runtime, Exception handling). A management of the 4 types of semanctics is required, if SaaS providers need to improve the SaaS application interoperability.

Self-evaluation for SaaS-applications analyzes ease of substitution of a service used by the by the SaaS application. At most of time, as SaaS applications use different services. Most of SaaS applications offer their functionalities via a combination of different services. Unfortunately, services provided can be unvailable for many reasons : Network unavailability, Failure of the provider of the service,etc. It is important that SaaS providers prevail other options just in case of unavailability of a used service. For high interoperable SaaS applications, the substitution can be automated. Ability of easy substitution of services by invoking similar services from other providers demonstrates a high level of interoperability. In order to evaluate their SaaS application, the SaaS providers assess that ease on the following scale : impossibility of substitution - Not all easy - Slightly easy - Moderately easy - Very easy.

Another interesting aspect of self-evaluation of SaaS application is the ability of SaaS applications to run on different cloud systems and use different ressources from many providers. Self-evaluation of SaaS applications analyzes those 2 different aspects. The ability of a SaaS application to be run on multiple disparate cloud providers involves a great capability of interoperability between the SaaS application and cloup systems.This means simply that components are deployed on different systems. Indeed, components will interoperate in order to achieve their tasks. Self-evaluation process assess at the same time the two aspect and classifies SaaS applications from those that can can only run on only one cloud provider and uses resources from one cloud provider to those that can multiple disparate cloud providers and uses resources from heterogenous cloud providers.

5.5 Application programming interface(API)

Used APIs should be compatible with multiple environments, or should have multiple independent implementations. The programming languages and the frameworks used to develop the application have to be enough flexible. Runtime-libraries could be available and compatible to the destination environment. SaaS providers should pay a special attention to the used technologies if they want their application the most interoperable as possible.

The openess of APIs that are used is very important in order to achieve SaaS interoperability. SaaS providers have to care about the degree of openess of the APIs they use. The choice for an open API will influent strongly interoperability of the SaaS applications within other systems. Open APIs are publicly available while closed(proprietary) APIs are private that implies that interfaces are only exposed to internal developers. Use of open APIs increases level of interoperability. Self-evaluation for SaaS applications includes the assessment of this scope. When SaaS applications use only proprietary APIs, they are less interoperable that those that use only open APIs. SaaS providers can chose to use both open APIs and propri-

etary APIs. In that case, the degree of interoperability of their SaaS applications will depend on the degree of the open APIs comparing to the degree of closed APIs.

The assessment of the degree of standardization of APIs used by the SaaS applications is an important aspect of self-evaluation of SaaS applications. Standardized APIs allow easy integration of the SaaS application in third applications. A high interoperability level needs that the SaaS application offers a high standardized API. The API specifies how data are exchanged and how services are invoked. The degree of standardization of the APIs shows how interoperabble is the SaaS applications. An API will be highly standardized if it can allow dynamic service discovery and dynamic service invocation. The standardization effort and the probable automation of interoperability resulting from standardization are part of self-evaluation of SaaS applications.

Following openess and standardization of used APIs, substitution of APIs is a dimension characterising interoperability of SaaS applications. Self evaluation of SaaS application assesses the ease of substitution of APIs. If the used APIs are easily substituable without bad side effects on interoperability of the SaaS application, that shows a good management of SaaS interoperability. A SaaS application that covers that point is better assessed.

Chapter 6

Tool

6.1 Methodology

The aim of this tool is the self-evaluation of SaaS applications. SaaS providers are the target of this self-evaluation. The main idea behind this effort is to examine the self-evaluation of SaaS applications. As the SaaS applications are wide, the tool goes beyond this multiplicity of SaaS applications, and attempts to focus on aspects that cross the diversity SaaS applications. The objective of the tool is to allow SaaS providers the ability to evaluate their applications, comparison between SaaS applications and aggregation of SaaS applications in clusters.

This approach includes two main parts : a state of the art about the self-evaluation of SaaS applications and the implementation of a tool for self-evaluation of SaaS applications. State of the art is based on litterature review. The litterature review shows that many approaches for evaluation of SaaS applications chose SaaS consumers as the target, we chose to adress this tool to SaaS providers. What is important is that through analysis of their SaaS applications, SaaS providers analyze indirectly themselves. By this way, the tool is very important at for the SaaS providers.

Research about methodology shows that there are mainly 3 great types of approaches : qualitative approach, quantitative approach and mixed approach. Qualitative approach uses open-ended questions. This is an emerging approach which deals with text or image data. Quantitative approach uses closed-ended questions. This is a predetermined approaches that processes with numeric data. And mixed approach uses both open and closed ended questions This approach is both an emerging and predetermined approach and deals with both qualitative and a quantitative data and analysis [11]. A quick view of the different approaches suggests the use of a quantitative approach. In effect, the aims the the self-evaluation incorporates measuring, scoring, ranking, comparison, aggregation. These tasks are feasible when we use numeric data.

As the the target of this self-evaluation involves the SaaS providers and given the fact that SaaS applications are numerous and cover many domains, the idea of using a survey covers many aspects. The questionnaire allow a good coverage of transverse aspects. The questionnaire are much used in the domain of evaluation, especially when this involves people. Moreover, the covered aspects may vary a lot from one SaaS provider to another. At one point one can imagine an automated tool, but , as some aspects are not not automatically auditable, the questionnaire seems to be an appropriate method.

The questionnaire is a self-administered questionnaire. For this purpose, the manner of distribution of the questionnaire is in the form of an online survey. By this way, questions are distributed at a minimal cost and it takes a few time rather than using paper-based questionnaire. There many steps in the building of the tool. The questionnaire is made of a set of technical questions. Even if it the target is the SaaS providers, technical skills are needed in order to answer well the questions provided through the questionnaire.

In order of giving a meaningful sense, a choice of precise criteria has been done. To identify the criteria, the first step was the identification of the relevant aspects of the Self-evaluation for SaaS applications. This identification has been made based on the litterature review about SaaS evaluations. The litterature review shows mainly the needs of users when they use SaaS applications. Thus, portability, availability and interoperability have been selected. Many users of SaaS applications are worried about those 3 when they start using the SaaS applications. This is also a challenge for SaaS providers to guarantee that their SaaS applications have acceptable levels of portability, availability and interoperabilty.

For each criterion, the first step was to define it well. For this purpose, a review of litterature about the criteria showed the main chararacteristics that are concerned. There the main points that are covered by the criterion have been set up. The second step was to find how to find properly how to design the questions in a way that achieves the fixed objectives. The different aspects retained for the differents criteria are :

1. **Portability of SaaS applications** :
    - Application portability
    - Ease of the application deployment from development environment to runtime environment
    - Ease of migration of the application from a cloud provider to another laying on a different Operating System
    - Likelihood of data-lock-in
    - Standardization of data format

- Ease of data migration(same database models)
- Ease of data migration(different database models)
- Data portability

2. **Availability of SaaS applications** :

- Level of availability
- Redundancy of the components of the SaaS application
- Model of backup
- Frequency of backup
- Recovery time guaranty
- Geographical dispersion of backup sites
- Failure detections mechanisms
- Monitoring, failover
- availability experience

3. **Interoperability of SaaS applications** :

- Coverage of the different scenarios of interoperability
- Openess of the used APIs
- Standardization of the used APIs
- Ease of substitution of used APIs
- Capability of running the SaaS application on disparate cloud providers
- Ease of substitution of services invoked by the SaaS application
- Data exchange formats
- Interoperability level
- Interoperability experience

6.2 Questionnaire design and scoring

The questionnaire is built in order to allow SaaS providers to evaluate themselves their SaaS applications. For this purpose, a serie of questions have been designed. The questions are subdivided into 3 types :

- There are dichotomous questions, where the respondents answer Yes/No given the fact that the desired characteristic is present or not. Dichotomous questions are ranked 0 or 1 given the fact that the SaaS providers meet or not the required aspect.

- There are matrix questions, where dichotomous questions(Yes/No) are clustered into a matrix. Each matrix holds 4 dichotomous questions. The aim of matrix questions is the verification of a set of characteristics that form a specific aspect. Matrix questions are ranked as follow : the score varies between 0 and 4. The 4 questions of a matrix question are characteristics of a precise aspect. The answers show how the the aspect is covered.
- There are also multiple choice questions where the respondents answer by chosing an option among 5 proposed options. The chosen option is the one that matches their situation. The benefit of using these 3 types of questions is the facility of counting scores and ranking the answers of the respondents. Multiple choice questions have 5 options. The options are ordered by their importance. Multiple choice questions are ranked between 0 and 4. Options are ranked from the worst to the best.

Table 6.1: Scoring scheme : dichotomous question : Score comprised between 0 and 1

Answer	No	Yes
Score	0	1

Table 6.2: Scoring scheme : Matrix question : Score comprised between 0 and 4

Answer	No	Yes
Score 1	0	1
Score 2	0	1
Score 3	0	1
Score 4	0	1

Table 6.3: Scoring scheme : Multiple choice question : Score comprised between 0 and 4

Answer	Score
Option 1	0
Option 2	1
Option 3	2
Option 4	3
Option 5	4

The questionnaire is subdivided into 2 main parts :

- the first part is a short is about characterizing the SaaS providers. The main concern in this part is the identification of general information about the SaaS

provider and the SaaS application. The question ask about the experience of the SaaS provider, the number of employees, the number of users of the SaaS application, the business domain, etc. The questions are not scored. These questions are interesting for the aggregation.

- the second part is about the questions about the criteria. The different criteria interoperability, availability and portability are covered. Each criterion has a total of 10 questions. There are 1 question for height determination. In effect, SaaS providers do not give the same importance to a given criterion. Then there are and 9 questions that are ranked. The overall score is calculated by weighting the different given the chosen weight per criterion. For each criterion, the possible biggest score is 33 .

Once the scores related to criteria weighted, we can then compute a global score as the result of the addition of the scores obtained for the each of the 3 criteria.

6.3 Online survey tools discussion

Once the decision to set up the questionnaire online, the next step was to find a service that allows to design online surveys. The researched features of that service were mainly the following :

- Possibility of automatic score calculation : as the respondent are answering to the questions, score may be computed at the same time
- Possibility of giving individual feedbacks to respondents : respondents should be given a score and a feedback based on the score
- Possibility of comparison between respondents : respondents should be given an indication about the ranking of the SaaS application
- Possibility of customization : configuration of CSS style, custom logos, colors, etc.

Among the tools available online, 5 have been are potential candidates :

Table 6.4: Comparison between online survey tools

Tool	auto scoring	individual feedback	comparison	customization
Google forms	KO	KO	OK	KO
Kwiksurveys	KO	OK	OK	OK
SurveyMonkey	OK	KO	OK	OK
QuestionPro	OK	OK	OK	OK
Fluidsurveys	OK	OK	OK	OK

After an analysis of the different features offered by the online tools described above, the choice has been raised to the google forms tool. Even if it seems to have

less features, the tools is free for use and there is no limitation in terms of number of questions. The other tools have yet included more advanced features. Despite the fact that google forms have less features, there are extensions that can be used to improve the tool.

6.4 Typology and Discussion

For the sake of a meaningful comparison between SaaS applications and SaaS providers, they are clustered in different groups. By clustering the SaaS applications in different groups, the process of Self-evaluation provides more information and has more significance. in the interest of comparison. For this purpose, 3 main characteristics have been retained :

1. *the number of employees that work for the SaaS provider* : SaaS providers are clustered following 4 groups given these levels.
   - Less than 10 employees
   - Between 10 and 50 employees
   - Between 50 and 250 employees
   - More than 250 employees
2. *the number of customers of the SaaS provider* : SaaS providers are clustered following 4 groups given these levels.
   - Less than 10 customers
   - Between 10 and 50 customers
   - Between 50 and 250 customers
   - More than 250 customers
3. *the experience of the SaaS provider* : SaaS providers are clustered following 4 groups given these levels.
   - Less than 1 year
   - Between 1 year and 5 years
   - Between 5 years and 10 years
   - More than 10 years

SaaS applications can also be clustered by business sector. Meanwhile, this aspect can vary a lot as SaaS providers given that there are many SaaS application developed for many business sectors.

The tool would allow to answer this serie of hypothesis:

- *Size of a SaaS providers affects the self-evaluation of the SaaS application* : Does the number of employees working for the SaaS providers improve or not the score of the SaaS application given the 3 criteria? We assume that more the size is, more the 3 criteria are covered.

- *Experience of a SaaS providers varies with self-evaluation of the SaaS application* : A great experience of SaaS providers allow them to be very specialized on a domain. Thus, we assume that more experienced SaaS providers will be interested in ensuring the respect of less criteria while the youngest SaaS providers will want to cover the 3 criteria.
- *Number of users of a SaaS providers varies with self-evaluation of the SaaS application* : A large number of users may indicate that the SaaS application have success with the users. However this does not mean that the 3 criteria have the same importance for different SaaS providers. We assume more we have a large number of users, more the 3 criteria will be covered.

The following charts represent a typology of a SaaS application. For each criterion, the chosen weight is 8 on the scale $0 - 10$. The criteria have the same weight.

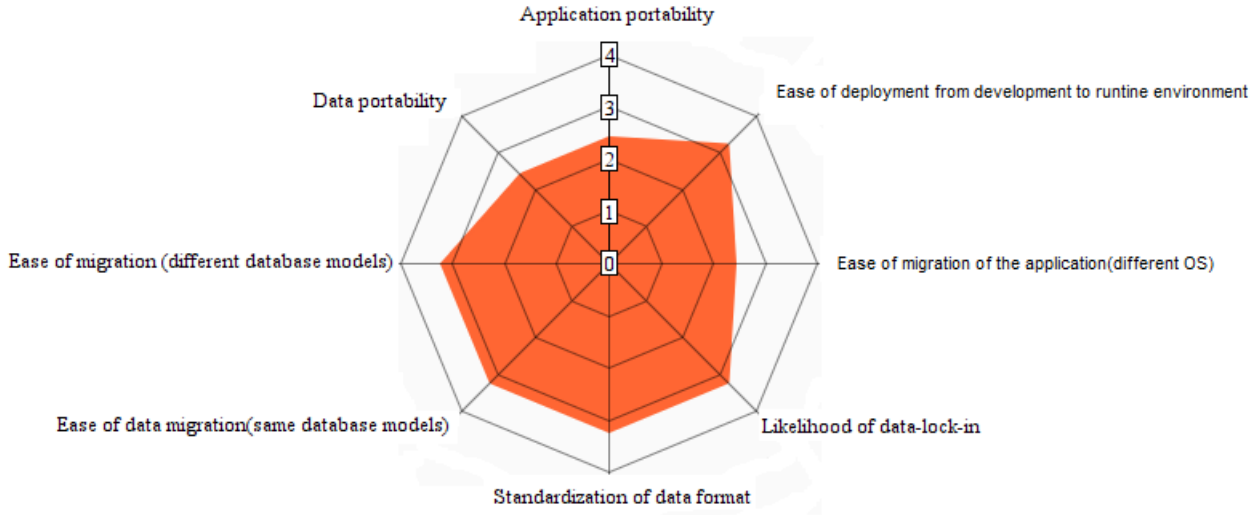


Figure 6.1: Typology : Portability

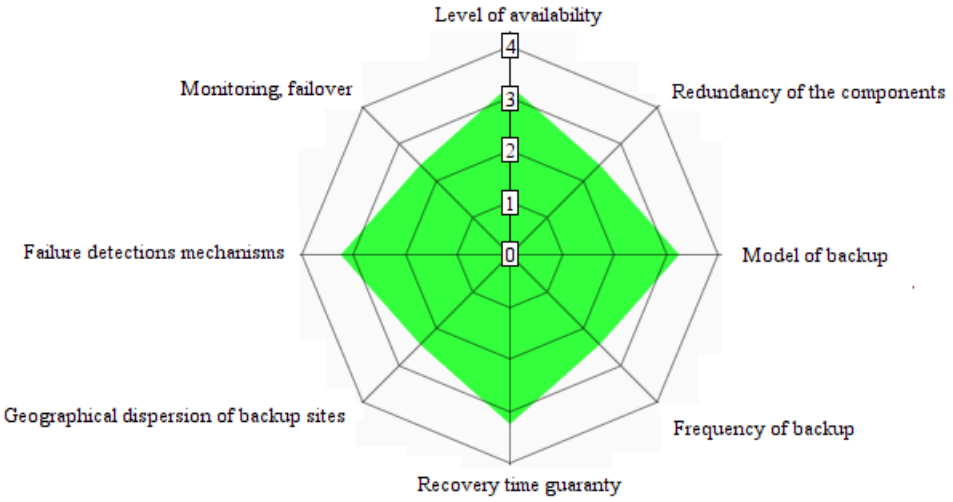


Figure 6.2: Typology : Availability

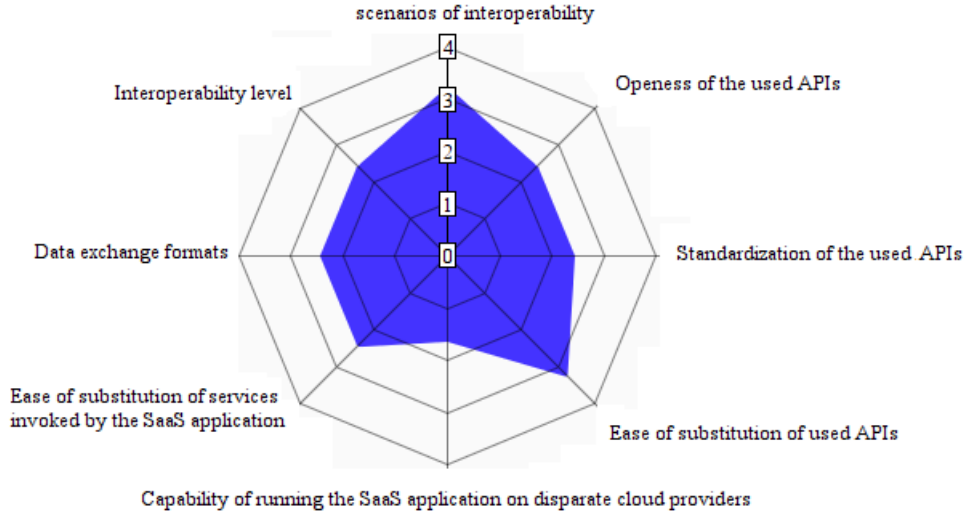


Figure 6.3: Typology : Interoperability

Chapter 7

Conclusion

This memoir was about the self-evaluation for Software as a Service applications. The idea behind was the implementation of a self-evaluation tool for Software as a Service applications. Our approach has followed different steps and has started by a state of the art about self-evaluation. In the first instance we have defined what is self-evaluation, and we have analyzed what is done in the domain of the Software as a Service. This part has been achieved through a deep litterature review more generally about self-evaluation and more generally about Self-evaluation of SaaS applications.

On the basis of that litterature review, we have then set up a methodology for implementing a tool for self evaluation of SaaS applications. Through the different methods of self-evaluation we choose to address a questionnaire to the SaaS Providers. By using a questionnaire, SaaS providers can see the improvement of their SaaS applications. They can themselves identify where they have to do improvement. As the questionnaire to be distributed as an online Survey the SaaS providers can also compare their applications with applications from other SaaS providers. Then they compare indirectly themselves with other SaaS providers.

Our litterature review has led us to establishment of a set of 3 criteria that are the portability, the availability and the interoperability of SaaS applications. The 3 have been defined and described and for each criterion we have chosen interesting aspect that need to be the baselines of the self-evaluation. Every aspect has been scanned in order to find the main characteristics that allow to self-assess the SaaS applications. The analyzes of each characteristic has been the basis of implementing the system of scoring that enabling ranking and comparison between SaaS applications and SaaS providers.

Besided those foregoing aspects, the self-evaluation tool for SaaS application allows us to depict a typology of SaaS application. Following the weight chosen by SaaS providers themselves, a landscape of SaaS applications can be painted. The 3 chosen criteria are weighted differently following the needs of each SaaS provider. The Self-evaluation tool can help us to see if the importance attached to each criterion

is consistent with the score obtained by a SaaS provider.

A possible improvement of the tool could be an addition of other criteria. By this way, the tool would cover many different aspects of the SaaS applications. Another aspect would be an introduction of qualitative aspects in order to supplement the quantitative aspect. Thus, it would give much more information about the SaaS applications landscape. This would open an opportunity for setting up the basis of a certification for SaaS applications and SaaS providers.

Appendix

Self- evaluation tool for SaaS applications

Questionnaire

Welcome Dear respondent,

This questionnaire is made for Self evaluate your SaaS application. The questionnaire is organised as followiwng:

1. The first part is a series of general questions about the SaaS provider and the provided SaaS application
2. The second, third and forth parts are questions related to criteria. The criteria used for this questionnaire are portability, availability and interoperability of SaaS applications. You will have to indicate for each criterion the level of importance accorded to it. These levels will be used as weights and will be helpful to compute the score.

Thank you in advance.

About you

1. **Name of the company***: _____
2. **Location of the company***: _____
3. **Your email ***: _____

4. **How many employees work for the company?***
   - ○ Less than 10 employees.
   - ○ Between 10 and 50 employees.
   - ○ Between 50 and 250 employees.
   - ○ More than 250 employees.
5. **Before the company begins its activities as a SaaS provider, was it yet working in the software sector?***
   - ○ No.
   - ○ Yes.
6. **How many customers the company has?***
   - ○ Less than 10 customers.
   - ○ Between 10 and 50 customers.
   - ○ Between 50 and 250 customers.
   - ○ More than 250 customers.
7. **What is the name of the SaaS application?***: _____
8. **What is the business sector?***
   - ○ Customer Relationship Management.
   - ○ Human ressources.
   - ○ Procurement.
   - ○ Communication.
   - ○ Other : _____
9. **How long the SaaS application has been used?***
   - ○ Less than 1 year.
   - ○ Between 1 year and 5 years.
   - ○ Between 5 years and 10 years.
   - ○ More than 10 years

Portability

10. **May you indicate on this scale how important is the portability of the SaaS application?***:
    1 ○—○—○—○—○—○—○—○—○—○ 10
    Application portability*

11a. **The application has yet been moved from on-premise environment to cloud environment** ○ No ○ Yes _____

11b. The application has yet been moved from cloud environment to cloud environment ○ No ○ Yes _____

11c. The application has yet been moved from cloud environment to on-premise environment ○ No ○ Yes _____

11d. It is possible to add, remove and configure components dynamically on the fly ○ No ○ Yes _____

12. How long did the application deployment from development environment to runtime environment take?*
Development environment and deployment environment may be different. A quick and easy or a heavy migration from development environment to runtime environment indicates the level of the application portability.

- ○ More than a day
- ○ Between 12 hours and a day
- ○ Between 6 hours and 12 hours
- ○ Between 1 hour and 6 hours
- ○ Less than 1 hour

13. How light would be the move of the application from the current PaaS/IaaS provider to an other one laying on a different OS?*
Programming languages need some runtimes and libraries. To ease portability, developers need to use standardized runtimes and libraries. But there can be great differences between OS, runtimes, libraries.

- ○ Major part of the sources would have to be rewritten and missed runtime libraries be imported
- ○ Minor part of the sources would have to be rewrittenand missed runtime libraries be imported
- ○ The code source would have to be recompiled and missed runtime libraries must be imported
- ○ The code source would have only to be recompiled
- ○ The code source is runnable immediately without any modification, recompiling or importing runtime libraries

14. **How likely is data-lock-in?***
Cloud providers(IaaS,PaaS,...) can provide to SaaS providers multiple ways to build their applications. Data lock-in depends on the level of openess of tools offered by the provider.
    - ○ My provider(s) only provide(s) proprietary tools for online development via a Web browser, using visual interfaces and design templates
    - ○ My cloud provider(s) offer(s) more custom functionality via native APIs than the possibility of deployment and development of arbitrary source-code
    - ○ My cloud provider(s) offer(s) custom functionality via native APIs as the possibility of deployment and development of arbitrary source-code
    - ○ My cloud provider(s) offer(s) less custom functionality via native APIs than the possibility of deployment and development of arbitrary source-code
    - ○ My cloud provider(s) only adopt(s) or provide(s) support for standards widely adopted and used technologies
15. **How standardized are the data formats used by the SaaS applicaton?* To ease data migration to other providers(IaaS,PaaS), data must be in standardized formats.**
Cloud providers(IaaS,PaaS,...) can provide to SaaS providers multiple ways to build their applications. Data lock-in depends on the level of openess of tools offered by the provider.
    - ○ Data formats have to be fully modified and automatic migration is impossible to achieve
    - ○ Data formats have to be partially modified and automatic migration is hard to achieve
    - ○ Data formats have to be less modified to allow semi-automatic migration
    - ○ Data formats do not need to be modified and semi-automatic is possible
    - ○ Data formats do not need to be modified and fully automatic migration is possible
16. **How easy is for a customer to migrate data used by your SaaS application to another one(same database models)?***
    - ○ Such case has never been met
    - ○ Not all easy
    - ○ Slightly easy
    - ○ Moderately easy
    - ○ Very easy

17. How easy is for a customer to migrate data used by your SaaS application to another one(different database models)?*

- ○ Such case has never been met
- ○ Not all easy
- ○ Slightly easy
- ○ Moderately easy
- ○ Very easy

Data portability*

18a. The cloud provider(IaaS, PaaS) offers capabilities of moving easily data ○ No ○ Yes _____

18b. The data formats are standardized ○ No ○ Yes _____

18c. The application is compatible with different database models(relational databases,NoSQL databases,...) ○ No ○ Yes _____

18d. The data have been yet moved from one environment to another one ○ No ○ Yes _____

19. Have you already experimented portability by moving the entire application, some of its components or data from one cloud provider to another one?*

- ○ Yes
- ○ No

Availability

20. May you indicate on this scale how important is the availability of the SaaS application?*:
1 ○—○—○—○—○—○—○—○—○—○—○ 10

21. What is the level of availability do you guaranty in your SLA?*
Given the kind of SaaS applications, different levels of availability can be required. There are some applications than need more availability than others.

- ○ Less than 90%
- ○ Between 90% and 95%
- ○ Between 95% and 97%
- ○ Between 97% and 99%
- ○ More than 99%

22. Are the components of your SaaS application redundant?*
To meet the level of availability, SaaS applications or some of their components may need to be redundant.

- ○ The components of my application are not redundant
- ○ Minimal components are redundant
- ○ Only the crucial components are redundant
- ○ Most of the components of the application are redundant
- ○ All the components of my application are redundant

23. In order to ensure your availability and the redundancy, what is the model of backup are you using?*
It is needed to replicate data to assure availability of an application. For that, many backup models exist, each backup model offering a different level of availability. Hot standby, requires a second data center that can provide availability within seconds or minutes, Cold standby, recovery requires hardware, operating system and application installation and Warm standby is a tradeoff between a hot and cold site

- ○ No secundary node
- ○ Hot standby
- ○ Warm standby
- ○ Cold standby
- ○ Active-Active (Load Balanced)

24. How oftendo you organize complete backup?*
Backup frequency is important for assuring availability. For example, critical applications need continous backup. Then, backup frequency indicates level of availability of SaaS applications.

- ○ Monthly
- ○ Weekly
- ○ Daily
- ○ Hourly
- ○ Continuously

25. How quick do you guaranty your recovery time?*

- ○ Never tested that
- ○ Recovery time is guaranteed in more than 24hours
- ○ Recovery time is guaranteed in 12-24hours
- ○ Recovery time is guaranteed in 1-12hours
- ○ Recovery time is estimated in less than 1 hour

26. How far are the different backup sites?*
It is necessary to separate geographically your backup sites. In case of disaster, having separated backup sites protects against outages

- ○ The backup sites are located at the same place
- ○ The backup sites are located at different places in the same region
- ○ The backup sites are located in different region
- ○ The backup sites are located in different countries
- ○ The backup sites are located in different continents

Failure detection mechanisms*

Having fault and failure detection at different levels of the cloud stack increase time for detecting failure. Quick you detect failures, quick you can resolve outages.

27a. Failure detection mechanisms are implemented at the application level ○ No ○ Yes _____

27b. Failure detection mechanisms are implemented at the virtual machines level ○ No ○ Yes _____

27c. Failure detection mechanisms are implemented at the infrastructure level ○ No ○ Yes _____

27d. The disaster recovery plan has been yet tested ○ No ○ Yes _____

Monitoring and Failover*

28a. The application users are offered monitoring capabilities ○ No ○ Yes _____

28b. The current state of the IaaS/PaaS provider are constantly monitored ○ No ○ Yes _____

28c. The Disaster Recovery Plan has been yet tested ○ No ○ Yes _____

28d. The failover mechanism has been yet tested ○ No ○ Yes _____

29. Has the application experienced yet outages or downtimes?*

- ○ No
- ○ Yes

Interoperability

30. May you indicate on this scale how important is the interoperability of the SaaS application?*:
1 ○—○—○—○—○—○—○—○—○—○ 10

Scenarios of interoperability*

SaaS applications can interoperate with other applications. These can be deployed in

the same cloud provider, in different providers but the same kind(homogenous) or in different providers of different kind (heterogenous). More scenarios you support, high is your level of interoperability

31a. The SaaS application can interoperate with other on-premise systems ○ No ○ Yes _____

31b. The SaaS application can interoperate with other SaaS systems within a cloud provider ○ No ○ Yes _____

31c. The SaaS application can interoperate with other SaaS systems in homogeneous cloud providers ○ No ○ Yes _____

31d. The SaaS application can interoperate with other SaaS systems in heterogeneous clouds ○ No ○ Yes _____

32. How open are the APIs used to develop the application?*
Open APIs are publicly available while closed(proprietary) APIs are private that implies that interfaces are only exposed to internal developers. Using open APIs increases level of interoperability.

- ○ The application only uses proprietary APIs
- ○ The application uses more proprietary APIs than open APIs
- ○ The application uses proprietary APIs as open APIs
- ○ The application uses more open APIs than proprietary APIs
- ○ The application only uses open APIs

33. Does the SaaS application any standardized APIs to allow its integration in third applications?*
A high interoperability level needs that the SaaS application offers a high standardized API. The API shoud specify how data are exchanged and how services are invoked. An API will be highly standardized if it can allow dynamic service discovery and dynamic service invocation.

- ○ The application doesn't offer any API and can only be used by the final users
- ○ The application offers APIs which are not standardized
- ○ The application offers APIs which are slightly standardized application offers standardized APIs that allows semi-automatic interoperability
- ○ The application offers highly standardized APIs that allows automatic interoperability

34. How easy is the substitution of the used APIs by other APIs?*
Easy substitution of used APIs shows how interoperable is an application.

- ○ Not all easy
- ○ Slightly easy
- ○ Moderately easy
- ○ Very easy
- ○ Extremely easy

35. Does the SaaS application capable to be run on multiple disparate cloud providers?*
A SaaS application that can run on multiple cloud providers need a high level of interoperability. Indeed, components need to interoperate to achieve their tasks. This indicates the degree of internal interoperability.

- ○ My application can only run on only one cloud provider and uses resources from one cloud provider
- ○ My application can only run on only one cloud provider and uses resources from heterogenous cloud providers
- ○ My application can run on multiple disparate cloud providers but uses resources from one cloud provider
- ○ My application can run on multiple disparate cloud providers but uses resources from homogenous cloud providers
- ○ My application can run on multiple disparate cloud providers and uses resources from heterogenous cloud providers

36. How is it easy to subsitute a service needed by the SaaS application by another one from another SaaS provider?*
SaaS applications offers their functionalities by combining services. As a SaaS provider, you can support yourself all the services needed by your application or you can invoke services from other SaaS providers. To be able to substitute easily each service by using services from other SaaS providers demonstrates a high level of interoperability.

- ○ Such case has never been met
- ○ Not all easy
- ○ Slightly easy
- ○ Moderately easy
- ○ Very easy

37. How standardized are data exchange formats to allow dynamic interoperability with random SaaS provider?*
To ease dynamic interoperability between SaaS providers, data must be in standardized formats.

- ○ Data formats have to be fully modified and automatic data exchange is impossible to achieve
- ○ Data formats have to be partially modified and automatic data exchange is hard to achieve
- ○ Data formats have to be less modified to allow semi-automatic data exchange
- ○ Data formats do not need to be modified and semi-automatic data exchange is possible
- ○ Data formats do not need to be modified and fully automatic data exchange is possible

Interoperability level*

38a. The SaaS application can easily exchange information with other SaaS applications ○ No ○ Yes _____

38b. The SaaS application supports many interoperability protocols and programming languages ○ No ○ Yes _____

38c. The SaaS application offers a well standardized API to allow integration in different systems ○ No ○ Yes _____

38d. The SaaS application only uses standardized and open APIs ○ No ○ Yes _____

39. Does the SaaS application offer APIs for allowing easy integration into third applications?*

- ○ No
- ○ Yes

Bibliography

[1] *Evaluation Thesaurus*. SAGE Publications, 1991.

[2] *Software engineering*. Pearson, March 2010.

[3] Asheesh Chaddha A V Parameswaran. Cloud interoperability and standardization. *SETLabs Briefings*, 7(7), 2009.

[4] Marjan Gusev Aleksandar Bahtovski. Analysis of cloud portability. *The 10th Conference for Informatics and Information Technology (CIIT 2013)*, 2013.

[5] Johannes Lipsky Stefan Tai Philipp Offermann Alexander Lenk, Michael Menzel. What are you paying for? performance benchmarking for infrastructure-as-a-service offerings. *Conference paper*, January 2011.

[6] Marvin C. Alkin. Debates on evaluation. *Newbury Park, Sage publications*, 1990.

[7] Robert Benefield. Agile deployment: Lean service management and deployment strategies for the saas enterprise. *Proceedings of the 42nd Hawaii International Conference on System Sciences*, 2009.

[8] Robert Benefield. Aized amin soofi, m. irfan khan, ramzan talib, umer sarwar. *International Journal of Computer Science and Mobile Computing*, March 2014.

[9] Lily Sun Chekfoung Tan, Kecheng Liu1. A design of evaluation method for saas in cloud computing. *OmniaScience Journal of Industrial Engineering and management*, February 2013.

[10] VNI Cisco. Forecast and methodology, 2013–2018.(2014).

[11] John W Creswell. *Research design : Qualitative, quantitative, and mixed methods approaches*. Sage publication, 2013.

[12] Athanasios V. Vasilakos Dana Petcu. Portability in clouds : Approaches and research opportunities. *Scalable computing : practice and experience*, 2014.

[13] DOINA DANAIATA and CALIN HURBEAN. Saas–better solution for small and medium-sized enterprises. *Environment*, 15(8.03):6–16, 2010.

[14] Plamena Zlateva Dimiter Velev. A feasibility analysis of emergency management with cloud computing integration. *International Journal of Innovation, Management and Technology*, April 2012.

[15] Mohammad Douglah. Developing a concept of extension program evaluation. *Series G3658-7. University of Wisconsin-Extension*, 1998.

[16] Dimitrios Kourtesis Fotis Gonidis, Iraklis Paraskakis. Addressing the challenge of application portability in cloud platforms. *7th South-East European Doctoral Student Conference*, 2012.

[17] Dimitrios Kourtesis Anthony J. H. Simons Fotis Gonidis, Iraklis Paraskakis. Cloud application portability: An initial view. *Proceedings of the 6th Balkan Conference in Informatics, ser. BCI '13*, 2013.

[18] P.JhansiRani Gangalam Swathi, M Vamshi Krishna. Survey on cloud computing services and portability. *IPASJ International Journal of Computer Science (IIJCS)*, May 2014.

[19] Joe O'Hara Gerry McNamara. The importance of the concept of self-evaluation in the changing landscape of education policy. *Studies in Educational Evaluation*, 2008.

[20] V.Sreenivas K.Bhavya, K.Yamini. Cloud services portability for secure migration. *International Journal of Computer Trends and Technology (IJCTT) - volume4Issue4*, April 2013.

[21] Won Kim. Cloud computing: Today and tomorrow. *Journal of object technology*, January-February 2009.

[22] Cheng J. C. McGibbney L. Kumar, B. Cloud computing and its implications for construction it. *Proceedings of the international conference in computing in civil and building engineering*, 2010.

[23] Sasko Ristov Kiril Kiroski Magdalena Kostoska, Marjan Gusev. Cloud computing interoperability approaches – possibilities and challenges. *In BCI(Local)*, 2012.

[24] Rean Griffith Anthony D. Joseph Randy Katz Andy Konwinski Gunho Lee David Patterson Ariel Rabkin Ion Stoica Michael Armbrust, Armando Fox and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. *UC Berkeley Reliable Adaptive Distributed Systems Laboratory*, February 10, 2009.

[25] National Institute of Standards and Technology. Nist cloud computing standards roadmap. Technical report, U. S. Department of Commerce, July 2013.

[26] Yashwant K. Malaiya Omar H. Alhazmi. Evaluating disaster recovery plans using the cloud. *IEEE*, 2013.

[27] Oracle. Oracle crm on demand. *available on http://www.oracle.com/us/products/applications/crmondemand/index.html*.

[28] INC RACKSPACE US. Software as a service perceptions survey. *Available on http://c1776742.cdn.cloudfiles.rackspacecloud.com/downloads/surveys/SaaSSurvey.pdf*, March 2007.

[29] Srikumar Venugopal James Broberg Ivona Brandic Rajkumar Buyya, Chee Shin Yeo. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, December 2008.

[30] Bahman Rashidi, Mohsen Sharifi, and Talieh Jafari. A survey on interoperability in the cloud computing environments. *International Journal of Modern Education and Computer Science (IJMECS)*, 5(6):17, 2013.

[31] Sai Peck Lee Zeinab Shams Aliee Reza Rezaei, Thiam Kian Chiew. A semantic interoperability framework for software as a service systems in cloud computing environments. *Expert Systems with Applications 41 (2014) 5751–5770*, 2014.

[32] Mikko Rönkkö Tuomas Mäkilä, Antero Järvi and Jussi Nissilä. How to define software-as-a-service – an empirical study of finnish saas providers. *First International Conference, ICSOB 2010*, June 2010.

[33] O. Williamson. The economic institutions of capitalism: firms, markets, relational contracting. *Free Press, New York*, 1998.