Cloud Readiness Assessment of Legacy Application

Flavio Corradini, Francesco De Angelis, Andrea Polini and Samuele Sabbatini
School of Computer Science, University of Camerino, Via del Bastione 1, Camerino, Italy

Keywords: Cloud Assessment, Cloud Migration, Software Evaluation, Reengineering, Cloud Application Portability, Cloud Governance.

Abstract: Applications and services hosted in the cloud are increasing continuously. Cloud technology offers important perspectives (performance, high availability, elasticity) and it enables new business models. Unfortunately, this new paradigm faces unprecedent requirements not addressed in legacy application (multi-tenancy, scalability, etc.). This leads to complex re-engineering phases in order to to migrate existing software into a cloud environment. Before starting a migration, it is important to analyze the cloud compliance of the application, what to expect after the migration and the effort required to fulfill these expectations. This paper assesses a way to extract an index that describes the feasibility of the re-engineering. We test the metric with a real application that needs to be migrated to a private cloud.

1 INTRODUCTION

Migrate legacy application to the cloud is one of the biggest challenges that cloud paradigm has brought (Buyya et al., 2010). Although the concept of utility computing was introduced about fifty years ago (Parkhill, 1966) , it began to be a commercial need only in the early 2000s. The fact that this new paradigm is driven by commercial aspects and not from a real scientific study has led to the creation of different definitions (Vaquero et al., 2008) depending on the commercial context. NIST (Mell and Grance, 2011) provides the most used definition of cloud. The cloud is totally revolutionary in software development as foundries have been in the hardware industry (Armbrust et al., 2009). This model is completely detached from the past, but it results in some problems. One of the main being the ability to migrate legacy application developed with previous methodologies into a new environment and making them cloud compliant. This challenge is due to the fact that legacy application have been implemented with previous methods without taking into considerations concepts unknown until the advent of cloud (i.e. elasticity and scalability) (Menychtas et al., 2013). To migrate a legacy application in a cloud environment, it is necessary to update the application to exploit these new capabilities. To do this, it is necessary to evaluate the application to migrate how and where the application is to be evolved. On the other hand, as mentioned previously, the cloud is not a unique concept, and so this assessment should also take into account the technology used in the cloud infrastructure. This article presents the definition of a metric to evaluate the compliance of an application respect to a cloud environment. This proposed research is related to the Open City Platform project (OCP project) founded by the Italian Ministry (Ministero dellIstruzione, dellUniversita e della Ricerca) in the Smart Cities and Communities and Social Innovation initiative(OCP, 2014). This project aims to migrate the applications used by some Public Administration in an private cloud infrastructure. In this context, the metric will be based mainly on technological concepts, ignoring the change in business models nedeed in a migration to a public cloud. The presented metric will be based on the specific request of this environment, and it will also be portable in order to be applied in other contexts. In Section II, the state of the art is analyzed. In Section III, the metric is proposed. Section IV will focus on the results provided by the metric. In Section V the application context is presented. The paper finishes with a section of Conclusions.

2 RELATED WORK

These last years, the issue of cloud migration was faced by researchers and industrials and a quite variety of solutions were presented.

Di Biase (Biase, 2013) proposes a questionnaire

Corradini F., De Angelis F., Polini A. and Sabbatini S..
Cloud Readiness Assessment of Legacy Application.
DOI: 10.5220/0005443301190126
In *Proceedings of the 5th International Conference on Cloud Computing and Services Science* (CLOSER-2015), pages 119-126
ISBN: 978-989-758-104-5

to evaluate both organizational and application migration in order to identify which migration type can be applied. Hosseini et al. (Khajeh-Hosseini et al., 2012) propose a migration tool-kit that involves all decision making in order to evaluate the feasibility of the application. Related to our work the application assessment is a list of question divided in different areas. Vu et al. (Vu and Asal, 2012) proposes a methodology approach is presented in order to establish which step are needed in legacy application evaluation process.

ARTIST (Artist, 2014) and REMICS (Remics, 2014) are two projects very closed to the aim of the research herein. These projects are funded by the European Community, and they focus their aim on migration using Model Driven Engineering (OMG, 2014). Both projects aim to develop different tools of different part of the migration. REMICS ended in the 2013 and it focused the attention on the recovery, migration, validation and supervising processes of the migration itself. However this project did not cover challenges such as elasticity, multi-tenancy and other non-functional properties. ARTIST focuses on migrating legacy software written in Java and C. The project is still open and it tries to support the migration in every aspect. Strictly related to the purpose of this article, ARTIST presents a work (Alonso et al., 2013) strictly related to the purpose of this article, where the pre-migration phase of the project is proposed. The method used to elaborate the maturity of the software is a questionnaire that has to be answered by a person with a good knowledge of technical and businesses aspects.

The evaluation of legacy application is a business used also by big cloud infrastructure player. Company such as Ibm (IBM, 2014), Cisco (CISCO, 2014), VmWare (VmWare, 2014) and other (RedHat, 2014) (Rackspace, 2014) (Amazon, 2014) offer a self-assessment tool or whitepaper to evaluate the advantages to migrate the application in their cloud. However, the problem of these approaches is that they are based on closed proprietary tools that are not widely available; and they are often accompanied by expensive consultancy periods. The advantage of our proposed metric is to create an agile process in order to fill out complex questionnaires readily.

3 EVALUATION CRITERIA

This section will presents the metric to assess if a legacy application is cloud compliant. This metric analyzes a series of questions that are asked to the software engineer. These questions are used to analyze the current state of the application and the status that should be achieved by migrating to the cloud. For the realization, the following categories were taken into account: (a) Workload, (b) Application Type, (b) Component, (c) Loose Coupling, (d) Distributed application, (e) Security, (f) Multi-Tenancy and (g) Database. Each category was then divided into several sub-categories in order to be able to identify the level of applications cloud compliant relating to specific category.

3.1 Workload

To migrate an application from in-house environment to cloud, it is necessary to take into account the workload. In Cloud computing patterns (Leymann et al., 2013) presents 5 different workloads: (a) Static, (b) Periodic, (c) Once a life, (d) Continuously grow and (e) Elastic. This paragraph goes in details of each type of load will be presented.

An application with *static workload* does not take any advantage to be migrated into cloud. This is due to the elasticity concept. Indeed, a static workload needs the same resources over the time, this means that having the automation in the allocation and deallocation of resources is almost useless. The migration of application with *periodic workload* into cloud will exploit the concept of resources elasticity. On the other hand, this workload is often too easy to be predicted, so it is possible to avoid the cloud by providing the necessary resources to meet the peak load and this would lead to a waste of resources during other periods. *Once a life workload* consists in a static load with rare peak of resources utilizations. This It results to be more advantageous than periodic load due to the fact that the single peak cannot be predicted so if in-house solution is used it would be probable to remain without sufficient resources. *Continuously grow workload* nearly represents the optimal case in which cloud migration will adds many advantages. In this type of load, the necessary resources grow with time and an automation of resource allocation would bring many benefits. In a static environment (in-house), this type of load would result in many problems as there would be either a state of over-sizing of the allocated resources or a lack of resources when the load has exceeded its capacity. This then leads to a waste of money when the available resources are greater than the actual demand, and it lacks of reliability and performance when the required resources are greater than those actually available. The cloud instead, thanks to its elasticity, allows the resources provided to be exactly those needed. For that reason *Elastic workload* is the optimal load for which the migration to the cloud is essentially required.

3.2 Architecture Type

The number of layer by which is divided the application is very important. The layers are a logical division that separate the various application processes and make them independent. Obviously, this type of classification is related to the server part. The applications that are taken into account are the classic: (a) 1-Layer: Monolithic, (b) 2-Layer, (c) 3-Layer (or more), (d) client-server.

1-Layer application has no subdivision, stateless and stateful components are related to each other and then it becomes difficult to carry out policies of scalability. In *2-Layer application*, it is possible to identify a data layer and a Presentation & Logic layer. This division helps the migration phase due to the fact that a division between stateful and stateless components has already been executed. Thus, the top layer can scale without any problem and not having to deal with the data redundancy. Applications with *3 or more layer* have a physical and logical subdivision already well defined. Each layer is able to be independent and satisfy a given operation. In a cloud environment, this subdivision allows a migration faster than the other cases giving the chance to each layer to scale. *Client-Server application* is another common architecture used in legacy application. This type of architecture has different problems when it is migrated into the cloud. In the cloud model the client part of the application has to be converted to work in a cloud infrastructure. However, if the migration affects only the server, this type of application could be seen as one of the previous specification.

3.3 Component

The analyses of components that would be migrated is an important aspect to be considered when a migration in cloud is performed. With the term *compoment* we consider a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard. Regards this category three different components are take into account herein: (a) Stateful with Strict Consistency, (b) Stateful with Evenutal Consistency, (c) Stateless.

Stateful with Strict Consistency components being the most difficult to migrate to cloud. The consistency must be guaranteed in all its replicas, thus the system must be able to keep synchronized all copies. The system needs a lot of work to fulfil this problem and when the number of replicas grows exponentially the performance of the system highly decreases.*Stateful components* can be accessed without having read the most updated data. It is not possible to use this model on critical components, which are the ones where information must be precise. That is why it is important to make sure data are as up-to-date as possible when read. In case of several instances, the updates are not executed synchronously but asynchronously, thus allows a better response. In a cloud view, this kind of component is able to scale much more easily than the previous one. *Stateless components* have no information that needs to be duplicated or updated in the other instances. This means it does not undergo efficiency loss as the number of replicas increases.n a cloud environment, the resources for components would be 100% exploitable, as they do not need any policy to be implemented and the updating of the data could be both asynchronous and synchronous.

3.4 Database

The database level is probably the most delicate to migrate. In order to achieve a good level of scalability the usage of NoSQL databases is recommended. However most legacy application uses Relational DataBase (RDB). Great might be the effort to migrate data from this databases to NoSQL, both in terms of to reengineering the database and migrate the data. This might result in discouraging this migration. Therefore, it is important to consider that components are only stateful at this level and data must remain intact. So, the issues to be considered are remarkable. The following list shows different database types: (a) relational database with stored procedure (SP), (b) RDB without SP, (c) RDB divided by area and (d) NoSQL.

Relational databases with Stored Procedures are the worst cases of migration to cloud. Until the advent of the cloud, using stored procedures was recommended as they were able to speed up and optimize the process. This also allowed to not have large deployment of resources for components that were not DB. With cloud and the theoretical availability of infinite and elastic resources, it is better to demand processing and workload to components able to scale without problems. Consequently, the part of the application more difficult to scale will be weighed down by stored procedures that load static components. As mentioned before, stored procedures are not recommended in cloud environment, due to the fact that they load components difficult to scale. *Without stored procedures*, the database must perform only the necessary operations on the data, without excessively overloading the database. All applications that use *multiple relational databases without stored procedure* apply to another sub-category of Database classification. The data layer of the application consists of multiple databases where each database has specific competences. In this case, the migration to cloud is better than in the previous cases since the size of the components different types of access are created, and the overhead of an area does not affect the performance of the other. *NoSQL databases* are the best for the migration to cloud. All non-relational database are included in this category. They are very useful because they can scale very easily and therefore the bottleneck that relational databases have is no longer present.

3.5 Loose Coupling

Loose coupling is another feature to be considered when an application is migrated to cloud environment is the component autonomy. A component is considered autonomous when its changes do not affect the other ones or vice-versa. With reference to the application elasticity in the cloud, components with a low level of autonomy make them difficult to divide, and then it becomes difficult to manage the scalability of a single component. Components with high autonomy can be scaled without affecting the other components with which it interacts. Starting from the 7 levels presented in (Krafzig et al., 2005), 5 levels of components autonomy were considered relevant during the migration: (a) Physical, (b) Format, (c) Time, (d) Reference, (e) Platform.

The highest level of coupling is the *Physical* one, in this configuration the migration in the cloud is very difficult to accomplish unless a review of the application is performed. A direct physical connection has a number of limitations. In this context, it is very complicated to undertake policies of scalability is since the increase of instances of a component implies major changes on how it interacts with the other components. *Format coupling* is intended for the components that are interfaced through a common format. The limitations are minor compared to a physical coupling carrying this specific type of applications on cloud has relevant advantages. However, this dependency leads to great limitations to the component.A component that does not have to be synchronized with other components falls into *Time cloupling*. The level of autonomy in this case starts to become significant, and the advantages resulting from the migration to cloud become clear. At this level, a component can exchange data with another one even if it is not available or it works at different speeds. In a cloud environment the component is easily scalable, thus achieving the benefits of scalability with little difficulty.A component is autonomous at a *reference level* when it does not need to know the address of other components to interact with them. In this case, the autonomy of the component is very high so moving component in a cloud environment does not create problems. With a *platform autonomy*, the component does not have any binding to the others, it can be implemented in any technology, and this would not affect the behavior of components around him. It is the best solution for a migration to cloud, each component is in fact completely independent on the other and it can perform all the operations without interfere with the other components.

3.6 Distributed Application

The migration on cloud of a distributed application can be easier that migrating other applications. Indeed, this means that Loose Coupling and Components concepts have already been taken into account. There are 3 different classification in distributed application: (a) pipe based through message, (b) process based, (c) layer based.

In *pipe based through message* distributed application, the division is made through the data. The components expect a certain input and provide certain output. Often, pipes and filters are used check the data format to ensure that the ”chain” between all components works. A *process based* distributed application is an application that focuses on decomposition based on business models. In this kind of applications, it is necessary to have an engine for the management of the processes, which manages each step of the application and ensures a proper work and the order of the components. A *layer based* distributed application decomposes the application into separate logical layers. Each layer is made of application components providing a certain function. Components are restricted to access components of the same layer or one layer below.

3.7 Multi-tenancy

Multi-tenancy is basis concept of cloud environment, that terms indicate the use of a single instance of the software by multiple tenants. The value defined to this category are: (a) multiple instance in separate hardware, (b) hardware in common with dedicated virtual machine for each tenant, (c) shared middle-ware with separated address space and multiple application instances and different db, (d) shared middle-ware with separated address space and multiple application instances and (e) shared middle-ware and one application instance.

In *multiple instance in separate hardware*, the multi-tenancy is not yet implemented. Each client organization has a dedicated stack from hardware to application level. In *hardware in common with dedicated virtual machine for each tenant*, the various tenants are located in the same physical machine in which VM were created specifically for each tenant. This configuration is not yet possible to be considered as multi-tenancy due to the fact that a large part of the stack is completely dedicated. *Shared middle-ware with separated address space and multiple application instances and different db* manages the multi-tenancy for a large part of the stack. Middle-ware in this case is in common, and only the application part is still runing at one instance per tenant. Although middle-ware is shared in this configuration, it is possible to see that the database uses different structures and tables depending on the tenant reference. *Shared middle-ware with separated address space and multiple application instances* is similar to the previous one, the only noticeable difference is in the management of data. In this case, the data of the various tenant reside on the same tables and there are no dedicated tables for each tenant. The rest remains unchanged, with middle-ware shared among multiple tenants and an application instance for each tenant. In *shared middle-ware and one application instance* the entire stack is managed through the multi-tenancy model. The tenant access to the same application instance and so the entire stack is shared. This is the best case for a migration to cloud because it makes the most out of this paradigm.

3.8 Security

Security is another key issue for cloud environment, even if in our context this problem is very mitigated. In our case, we are talking about private cloud where will run own applications. This differentiates our model from the classic scene where public cloud is taken into account. Despite the context fades this issue, it is always important to take security into consideration. In our case the security concerns basically two aspects: permissions and data protection. This is the only category that is divided in other categories due to the different meaning of security. The result of this category is the sum of the value of Authorization and Data Protection.

3.8.1 Authorization

The first aspect of security taken into account concerns the Permissions. Access management is very important in cloud because of the multi-tenancy, that, as explained above, must manage multiple tenants on the same instance. For this reason, the applications were classified as: (a) application without login, (b) application with simple login, (c) application with login managed by roles

3.8.2 Data Protection

The data protection is the second aspect to be considered in security. The classification in this case concerns the type of data, considering whether they are sensitive or not, and whether they are encrypted or not. This classification is due to the fact that having a shared environment requires special attention to data protection and it is extremely important for the data not to be violated by unauthorized users.

4 MIGRATION ASSESMENT

The objective of the metric is to provide to the user the cloud compliance of an application. To define a clear output, two modes were defined: a numeric output represented by three values, and a graphical output. To perform the assessment, for each category and subcategory we assigned a specific value (Table 1). The values assigned in this article have been set according to our infrastructure. However it can be changed depending on the needs of the cloud service provider that delivers the platform and the questionnaire of the assessment.

In order to calculate the metric, the users fills out a questionnaire of technical questions in order to define the positioning of the application in the various criteria. In addition, the questionnaire presents some questions to determine which would be the goal of the migration. Indeed, to exploit the potential of the cloud, it is not necessary that the application reaches the maximum value in each category to exploit the potential of the cloud. The metric provides values for the current state (equation 3) and the final one (equation 4) for each category. These extracted values are then processed according to the equations 5 and 6. These values represent the percentage of actual (equation 5) and future (equation 6) application compliance respect to the cloud provider (equation 5). These two values are used to extract the percentage of fulfilment of the desired objective (equation 7). This is the most important value of the metric. This value can be used to estimate the effort of the migration in terms of time. This calculation can be made taking into account the costs of the implementation of the software until now, and relate them to the index presented in equation 7. Although this effort is not predictable in a precise way, the index can help in the estimation when it is calculated for more that one

Table 1: Weight of Categories and Sub-Categories.

WORKLOAD		**10**
-	Unknown	0
-	Static	2
-	Periodic	6
-	Once a life	7
-	Continuously grow	9
-	Elastic	10
LOOSE COUPLING		**8**
-	Physical	0
-	Format	2
-	Time	4
-	Reference	7
-	Platform	10
NUMBER OF LAYER		**7**
-	No (1-Layer)	0
-	client-server	2
-	2-Layer	5
-	3+-Layer	10
DISTRIBUTED APPLICATION		**5**
-	No	0
-	Pipe Based through message	7
-	Process Based	9
-	Layer Based	10
DATABASE		**9**
-	RDB with SP	0
-	RDB without SP	4
-	RDB divided by area	7
-	NoSQL	10
COMPONENT		**8**
-	Stateful with Strict Consistency	2
-	Stateful with Evenutal Consistency	6
-	Stateless	10
MULTI-TENANCY		**9**
-	Multiple instances in different separate hardware	0
-	Hardware in common with VM for each tenant	2
-	Shared middelware, separated address space, multiple application instances, different DB	6
-	Shared middelware, separated address space, multiple application instances	8
-	Shared middelware and one application instance	10
SECURITY		**8**
-	AUTHORIZATION	**5**
	No Login	0
	Simple Login	2
	Login with specific role	5
-	DATA PROTECTION	**5**
	Non encypted sensitive data	0
	Non encrypted non sensitive data	2
	Encrypted sensitive data	4
	Encrypted non sensitive data	5

application, the underline implementation technology is comparable, and we know the effort made in other migrations.

$$n = \text{Number of Categories} \tag{1}$$

$$W_i = \text{Weight of the Category i} \tag{2}$$

$$V_{i,x} = \text{Current value of the Category i} \tag{3}$$

$$V_{i,y} = \text{Desidered value of the Category i} \tag{4}$$

$$V_1 = \frac{\sum_{i=1}^{n}(W_i \cdot V_{i,x})}{\sum_{i=1}^{n}(W_i \cdot V_{i,max})} \cdot 100 \tag{5}$$

$$V_2 = \frac{\sum_{i=1}^{n}(W_i \cdot V_{i,y})}{\sum_{i=1}^{n}(W_i \cdot V_{i,y})} \cdot 100 \tag{6}$$

$$V_3 = V_1 / V_2 \cdot 100 \tag{7}$$

A radar chart was chosen to represent graphically the results. The choice fell on this type of chart because it gives the possibility to have a rough assessment of the status of the application and to understand what are the gaps to be faced. The various categories are represented in each edge of the chart. By means of the questionnaire previously issued, each edge shows two values representing the score obtained by the application in a category. The first value is related to the current state of the application, while the second one refers to the value that should be achieved when migrating to the cloud. As mentioned above, each category has a different score, so before the chart is drawn, all the values are normalized according to the weight of the category. The difference lies in the effort that must be put in before the migration. Moreover, radar chart allows the increasing or decreasing of various edges without changing the meaning of it, that advantage permit to export the assessment in other environments only configuring the metric.

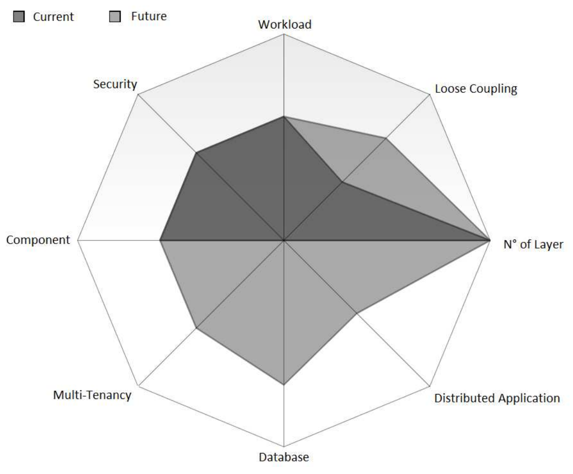


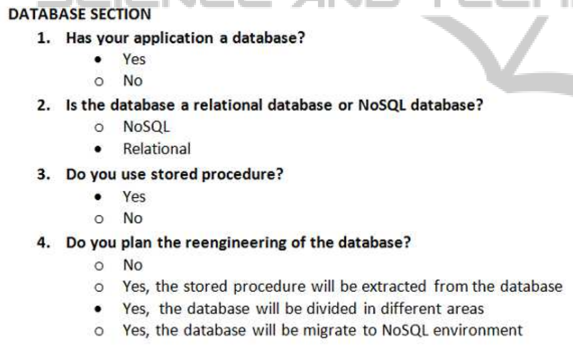
Figure 1: Example of radar results after the evaluation.

4.1 Metric Utilization Example

In this section we introduce an example of the metrics presented above using an application that handles the calculation of vehicle taxes.

4.1.1 Questionnaire

The questionnaire is composed of a set of questions with multiple answers, each answer is needed to extrapolate one of the value presented in the Table 1.The set of questions regarding ”database” criteria is presented in Fig. 2 as example. The question regards both current status of the application (i.e. Q1-Q2-Q3) and future status (i.e. Q4). When the questionnaire is completed the system elaborates the answer to select proper value. In example, Fig. 2 shows how the first three questions allow us to extrapolate the current value of the application (i.e. RDB with SP”, value =0) whereas the fourth one refers to the value the application would have after the migration (i.e. RDB divided by area, value=7).



DATABASE SECTION

1. **Has your application a database?**
   - • Yes
   - o No
2. **Is the database a relational database or NoSQL database?**
   - o NoSQL
   - • Relational
3. **Do you use stored procedure?**
   - • Yes
   - o No
4. **Do you plan the reengineering of the database?**
   - o No
   - o Yes, the stored procedure will be extracted from the database
   - • Yes, the database will be divided in different areas
   - o Yes, the database will be migrate to NoSQL environment

Figure 2: Database’s questionnaire section.

4.1.2 Results

In this section we present the results of the application tested. The results of the questionnaire are presented in Table 2(the database values are already explained in the previous section). These values are then used to calculate the indexes presented earlier. The results of the elaboration are: $\mathbf{V_1} = 32.25$ %, $\mathbf{V_2} = 56.125$ %, $\mathbf{V_3} = 57{,}461$ %. As said in the previous paragraph, V_3 is the important value. Indeed, this value can be used in particular application (such as homogeneous applications) to estimate the effort needed for the migration. Considering the effort used until now to achieve V_3, it is possible to estimate the effort necessary to complete the migration. The figure shown previously (Fig.1) represents the results of this elaboration in a graphical manner. Every edge has a maximum value fixed to 100 ($W_{max} \cdot V_{max}$), the value of the current application (dark grey) is $(W_i \cdot V_{i,x})$ (in the example $9 \cdot 0 = 0$) and the desired value (light grey) is $(W_i \cdot V_{i,y})$ (in the example $9 \cdot 7 = 63$).

Table 2: Questionnaire’s results of example application.

CATEGORY	CURRENT VALUE	FUTURE VALUE
Workload**(10)**	Periodic **(6)**	Periodic **(6)**
Loose Coupling**(8)**	Time **(4)**	Reference **(7)**
N Layer**(7)**	3+ -Layer **(10)**	3+ -Layer **(10)**
Distributed App.**(5)**	No **(0)**	Layer Based **(10)**
Database**(9)**	RDB with SP **(0)**	RDB divided by area **(7)**
Component**(8)**	Stateful with eventual consistency **(6)**	Stateful with eventual consistency **(6)**
Multi-Tenancy**(9)**	Multiple instances in different separate hardware **(0)**	Shared middelware, separated address space, multiple app. instances, diff. DB **(6)**
Security**(8)**	Simple Login + Encrypted sensitive data **(7)**	Simple Login + Encrypted sensitive data **(7)**

5 FUTURE WORK AND CONCLUSIONS

The next step will test the assessment other application involved in OCP project in order to better validate the index. Moreover, the presented metric is used to evaluate software engineering aspects. However, the adoption of cloud computing involves all the aspects of the enterprise. The aim is to integrate this evaluation metric with other metrics that evaluate business and organizational aspects. The integration with other metrics imply the automation of the metric in order to have a migration methodology to automate the entire process (Fig. 3). The idea is to use Model Driven Engineering to extract information directly from the artifacts in order to have as accurate information. In this way the questionnaire to submit will undergo to some changes. The defined weights will also be automated in order to have the assessment made automatically from a data storage in which patterns and services of the cloud infrastructure are described. The crucial aspect of this assessment is the value presented in equation 7 that is used to estimate the effort needed to the migration. The proposed estimation of effort uses generic assumption. The idea is to study in depth how it would be possible to improve the accuracy of the estimation effort by transforming it to person/month or other valuable metrics. Another output will generate documents describing the weakness of this evaluation. These documents will be used as an input by the modernization process of the application that will perform an evolution of the application even going to insert, where possible, specific patterns of the infrastructure to create optimal Platform Specific Model (PSM). The objective of these implementations is: (a) to have a precise and clear situation through automated processes, (b) to make the metric portable in other contextes.

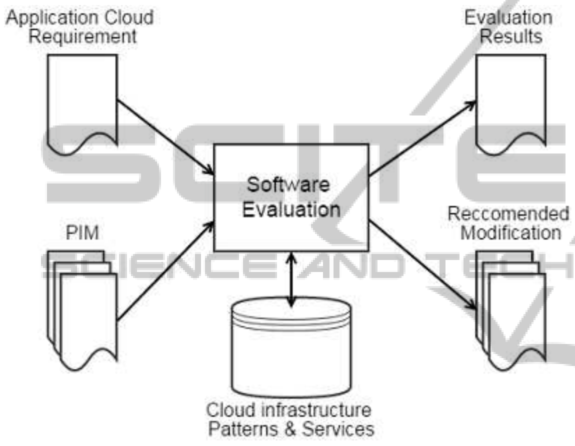

Application Cloud Requirement
Evaluation Results
Software Evaluation
PIM
Reccomended Modification
Cloud infrastructure Patterns & Services

Figure 3: High level architecture diagram of a migration phase related with the metric.

Public institutions suffer the issue of legacy application migration to the cloud. They offers a wide range of heterogeneous services to citizen, company and other institution and they could take advantage from cloud computing handling automatically and dynamically resources. The metric proposed in this paper is intended to helps public institution that wants to create private cloud infrastructure with services installed in it. Considering the context of the OCP project, in which this research is performed, it was decided to consider only the technological part of the migration ignoring aspects of business. At the same time, a platform independent setting was given not to limit the metric only to this case but to apply it in other areas.

ACKNOWLEDGEMENTS

This work has been supported by the Open City Platform project and EUREKA project funded by Regione Marche, UNICAM and APRA SpA.

REFERENCES

Alonso, J. et al. (2013). Cloud modernization assessment framework: Analyzing the impact of a potential migration to cloud. In *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*, pages 64–73. IEEE.

Amazon (2014). http://media.amazonwebservices.com/-cloudmigration-main.pdf.

Armbrust, M. et al. (2009). M.: Above the clouds: A berkeley view of cloud computing.

Artist (2014). Artist project. http://www.artist-project.eu/.

Biase, F. D. (2013). Legacy to cloud migration: Assessing the cloud readiness of legacy software systems. Master's thesis, University of Applied Sciences Western Switzerland.

Buyya, R., Ranjan, R., and Calheiros, R. N. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer.

CISCO (2014). . http://www.ciscowebtools.com/cloud.

IBM (2014). http://www-01.ibm.com/software/rational/info/cloud-services/self-assessment.htmls.

Khajeh-Hosseini, A., Greenwood, D., Smith, J. W., and Sommerville, I. (2012). The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. *Software: Practice and Experience*, 42(4):447–465.

Krafzig, D., Banke, K., and Slama, D. (2005). *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional.

Leymann, C. F. F., Retter, R., Schupeck, W., and Arbitter, P. (2013). Cloud computing patterns.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing.

Menychtas, A. et al. (2013). Artist methodology and framework: A novel approach for the migration of legacy software on the cloud. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*, pages 424–431. IEEE.

OCP (2014). Open city platform project. http://www.opencityplatform.eu/.

OMG (2014). Mda. http://www.omg.org/mda/.

Parkhill, D. F. (1966). Challenge of the computer utility.

Rackspace (2014). http://www.rackspace.com/cloud/hybrid.

RedHat (2014). https://engage.redhat.com/forms/cloud-readiness-assessment.

Remics (2014). Remics project. http://www.remics.eu/.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55.

VmWare (2014). http://www.cloudflightcheck.com/.

Vu, Q. H. and Asal, R. (2012). Legacy application migration to the cloud: Practicability and methodology. In *Services (SERVICES), 2012 IEEE Eighth World Congress on*, pages 270–277. IEEE.