An Automated Architecture for Portability in PaaS using Virtual Appliance

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Rajinder Sandhu
(801131020)

Under the supervision of:
Dr. Inderveer Chana
Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2013

Acknowledgement

My time as a postgraduate student at Thapar University, Patiala is supported by many kind people. This page attempts to recognize those who have been most helpful along the way.

First of all, I thank God for providing me such an opportunity and support.

I am deeply indebted to my supervisor, Dr. Inderveer Chana. Her advice, support and kind encouragement on thesis and professional issues have effectively guided me to the right path. She always led my research papers to clear, smart and effective ones. Without her this thesis would not be possible.

Thanks to all members of Thapar University. Dr. Maninder Singh, Head of Computer Science Department, for his kind support and directing me in field of research based on his extensive knowledge. All staff members of Computer Science and Engineering Department for providing me all the resources required for the completion of my thesis.

I would like to thank my friends, Amandeep Singh Sidhu, Amritpal Singh, Taranpreet Kaur, for their moral and technical support. They always encourage me when I needed the most.

Outside the Computer Science and Engineering Department, I benefited from time spent with friends especially Navjot Sandhu, Harkirshan Singh and Gurusharan Virk. Interdisciplinary discussion with these friends has given me many new ideas and perspectives to work on my thesis.

I want to express my appreciation to each and every person who directly or indirectly help me to complete my thesis. Finally, I thank my family for all aspects of life.

Rajinder Sandhu

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*An Automated Architecture for Portability in PaaS using Virtual Appliance*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Inderveer Chana* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Rajinder Sandhu

(Rajinder Sandhu)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Inderveer 11/07/13

(Dr. Inderveer Chana)

Associate Professor

Computer Science and Engineering Department

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Abstract

Over the past decade, with Cloud Computing, IT industry is advancing towards an era where resources are delivered as a service rather than a product. Cloud Computing provides applications, platforms, and resources delivered as a service over the internet with pay-as-use model. Cloud Computing is popular because it promises a return-on-investment, better scalability, dynamic provisioning, transparency, reduced capital cost, and zero maintenance and operational cost. Cloud Computing can be very beneficial for small-scale and mid-scale organizations because they can lease computer infrastructure at economical rates which reduces capital cost.

With the adoption of Cloud Computing by reputed organizations, it diverges into many standards and models which results in heterogeneity in the Cloud Environment. Heterogeneity between Cloud providers causes many problems like vendor lock-in, portability and interoperability, preventing future adoption of Cloud Computing for middle scale and small scale organizations. Interoperability and Portability among Cloud providers is the only solution to avoid vendor lock-in situation and a roadmap toward more competitive market for cloud providers and users. The focus of this research work is to solve portability conflict in PaaS offerings using virtual appliance.

This thesis presents an automated architecture for portability conflicts raised during deployment and migration of an application in a PaaS environment by separating three fundamental PaaS entities: Application, User Data and Infrastructure. Resource Discovery is a cumbersome process and often suffers due to interoperability issues. In the proposed architecture, this has been addressed through Semantics and Similarity Graphs. Semantics refers to study of meaning rather than syntax. User requirements are expressed semantically using ontology. Similarity Graphs utilize graph created from ontology and calculate similarity between two nodes by allowing numerical computation rather than syntax reasoning. Quality of Service (QoS) is maintained by signing a Service Level Agreement (SLA) between all parties and it is monitored by third party SLA manager. An algorithm is proposed for providing compensation in case of violation of stated SLA. Experimental results demonstrate the working of the proposed architecture and show how portability can automatically be achieved in a PaaS environment using virtual appliance.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

With the expansion of network bandwidth, many interesting uses of the internet are emerging in recent times. One of them is remote execution of task on distant and distributed physical machines commonly known as Grid Computing. Grid Computing originated in research in the mid-1990s. Grid Computing allows user to use idle resources on a network for a complex problem. After Grid Computing expanded, it paved way to a new type of computing which came into existence around the end of 2007 known as Cloud Computing [1]. Cloud Computing provides resources and applications, known as services, over the internet to the user. This chapter provides a brief overview of Cloud Computing by discussing its core characteristics, different service layers, application areas and open challenges. Motivation and organization of thesis are discussed at the end of this chapter.

1.1 Cloud Computing

With the development of parallel computing, distributed computing and grid computing a new computing evolved named as Cloud Computing. The main aim of Cloud Computing is to provide all IT resources as a service to end user whether it is storage, CPU power, software or anything. Cloud Computing virtually provides an infinite pool of resources which can be allocated to different users for short span with pay-as-use basis. Cloud Computing is very economical because it doesn't require any upfront commitment from Cloud users [2]. Figure 1.1 below shows stack of Cloud Computing applications. Physical hardware is at base level which is encapsulated using virtualization technologies, Cloud applications use these resources to create a Cloud service, Cloud services interact with the user to provide them their specified services.

Cloud is widely referred to many technologies such as Virtualization, Utility computing, IT outsourcing, and Platform as Services. So defining Cloud Computing is very difficult and complex. There are already more than 20 definitions of cloud computing [3]. According to Buyya et al. [4], a cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized

computers that are dynamically provisioned and presented as one or more unified computing resources based on service level agreement (SLA) by enabling ubiquitous, convenient, on-demand network access to a shared pool of computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

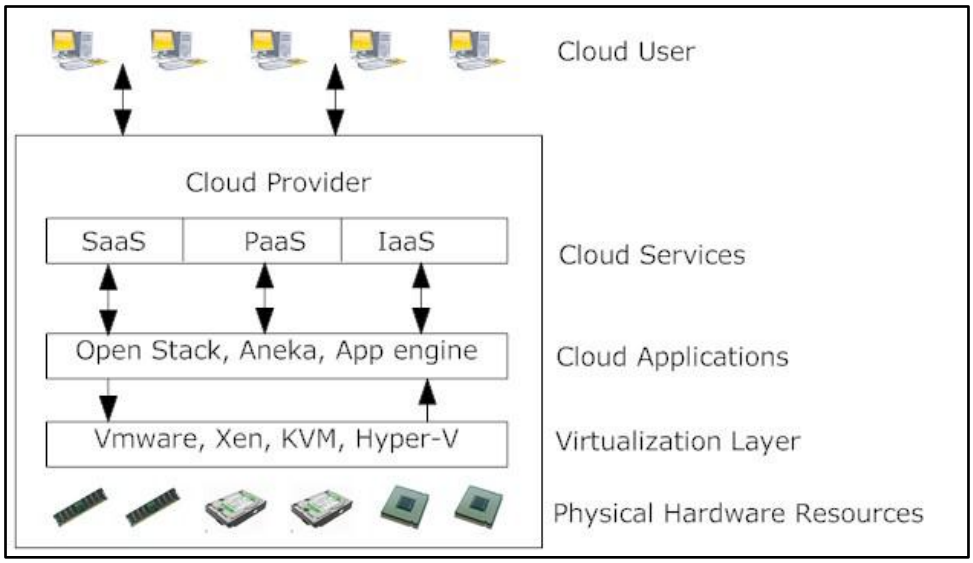

Figure 1.1: Stack of Cloud Computing

1.2 Characteristics of Cloud Computing

Some of the characteristics of Cloud Computing are explained below [5] [6] [2] [1]:

- Illusion of infinite resource available on demand- Cloud Computing provides virtually infinite number of resources for users on demand over the network. This increases agility in defining and structuring software systems. Since users can rent IT resources, they can compose their software systems for dynamic and flexible need. Capacity planning is reduced since Cloud Computing allows reacting to unplanned demand quite rapidly. All provisioning and releasing of IT resources is dynamic and automated so not require any human interaction.
- Elimination of up-front commitment- Capital cost is the cost associated to assets that need to be paid in advance to start a business activity. Before Cloud Computing capital cost are paid in advance to start the business. The revenue of the business is then utilized to compensate over time for capital cost. Thus, Cloud Computing significantly helps to increase the net gain by providing

resources without any up-front commitment. It also provides opportunity for small organizations to start a business and comfortably grow.

- Pay as you Use- The most evident benefit of Cloud Computing systems and technologies is the increase in economical return due to reduce maintenance and operational cost related to IT software and infrastructure. This is because IT assets, namely software and infrastructure, are turned into utility and money is paid until they are used for example Processor for an hour and storage for the day.
- Insurance for Quality of Service- With the introduction of Service Level Agreement (SLA) and SLA violation compensation algorithms quality of service delivered is maintained. Thus, the user doesn't have to worry about Quality because proper compensation will be provided if SLA is violated.

1.3 Cloud Computing Applications

Cloud is now used by most of the industries due to its benefit of dynamic scalability and no upfront commitment. This section introduces some successful implementation of Cloud Computing in scientific and productivity applications.

1.3.1 Scientific Applications

These types of applications require high computation power and throughput. Researcher in scientific field require very large amount of resources but for very short span of time. Cloud Computing serves best deal for these type of applications. Below some of successful applications of Cloud Computing in this field of work have been discussed.

- ECG Analysis in the Cloud- Continuous monitoring of health is new buzz word in the field of medicines. In this application Cloud is used to analyse patients Electro cardio Graph (ECG) and make predication about any heart diseases. An instrument is placed near to patient heart which continues monitor heart condition. Data from this instrument is received by mobile phone. Mobile phone forwards this data to Cloud which analyses this data [7]. Cloud is not necessary in this application but it provides two important benefits. First is scalability, infrastructure automatically scales up and down

according to requirement. Second is capital cost investment, hospital don't have to buy new resources in seasonal demands [8].

- Satellite Image Processing- Spatial and Non-spatial data is collected by satellites. This raw data requires very large amount of processing to make it useful to Geographic Information System (GIS). A Cloud based application is developed by Indian Space Research Organization (ISRO) to process these images [9]. ISRO use Aneka platform to develop an application which provides a set of functionality for Geocode generation and data visualization. Processing time is reducing considerably using Cloud environment [8].

1.3.2 Productivity Applications

Productivity applications use Cloud Computing for some commonly used desktop applications. Document storage, office automation, website deployment like tasks can be done with easy and effectively using Cloud. Below some of productivity applications using Cloud Computing are explained.

- Google App Engine- No need to instantiate any virtual machine. Application written in Python or Java can directly be deployed [10]. Google charge on the actual normalized CPU cycles used. Storage is only non-relational. Charge is calculated on these parameters – bandwidth, CPU, storage, emails send. Bandwidth usage charges are $0.12 per GB, CPU cycles usage charges are from $0.08 to $0.64 per hour depending upon the capacity, storage charges are $0.13 to $0.64 per GB per month [11]. $0.0001 are charged per email. Google provides free quota for each of these parameters – it may be enough for development, testing and small deployment. There are limits imposed for peak usage on many different parameters – with daily limits & limits on usage in a burst.
- Drop Box- Dropbox [12] is a file hosting service operated by Dropbox, Inc., that offers cloud storage, file synchronization, and client software. Dropbox allows users to create a special folder on each of their computers, which Dropbox then synchronizes so that it appears to be the same folder (with the same contents) regardless of which computer is used to view it. Files placed in this folder also are accessible through a website and mobile phone applications.

- RoboEarth- It is a European project led by the Eindhoven University of Technology, Netherlands, to develop a WWW for robots, a giant database where robots can share information about objects, environments, and tasks. Researchers at ASORO (A-Star Social Robotics Laboratory, Singapore) have built a cloud-computing infrastructure that allows robots to generate 3-D maps of their environments much faster than they could with their onboard computers [13].

1.4 Cloud Computing Service Layers

In this section, all three service layers of Cloud Computing are explained. Figure 1.2 shows major difference between classic approach, IaaS, PaaS and SaaS.

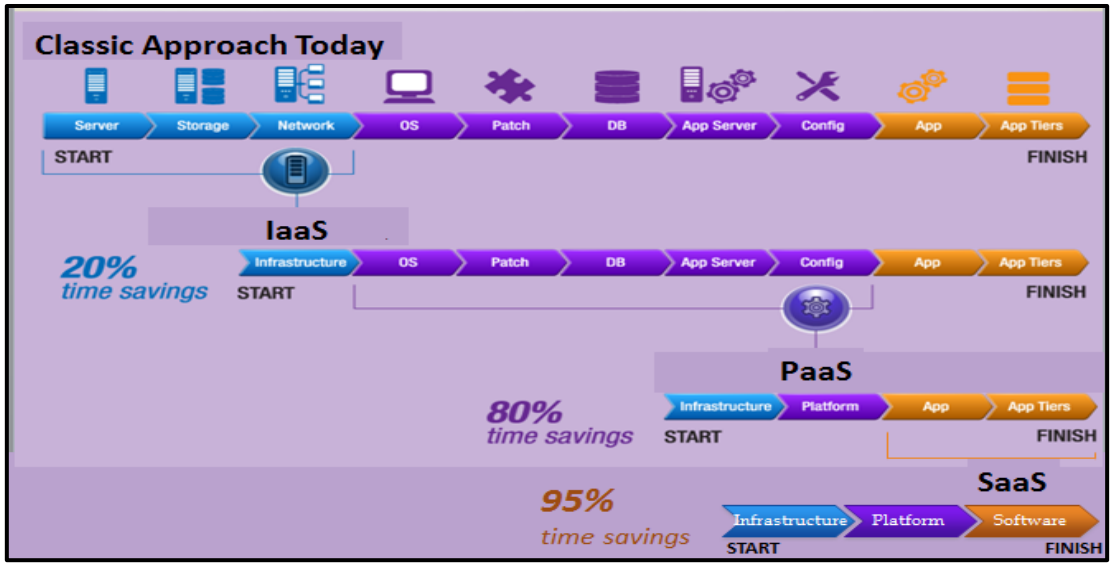

Figure 1.2: SaaS, PaaS, IaaS Configurations

As shown in Figure 1.2, in classic approach users have to manage all servers, network and storage need of software application. These requirements are directly provided by IaaS, users don't have to manage basic hardware but Operating system and Software installations has to configured. PaaS solves operating system installation problems by providing then as a service but software development is still to be done by Cloud user. At last, SaaS manage all up to application installation. User just uses the application and get results. 20% time can be saved using IaaS which can be extended to 80% and 95% in PaaS and SaaS respectively. This section explains all basics of IaaS, PaaS and SaaS [14].

- Infrastructure as a Service- Infrastructure as a Service (IaaS) solutions are the most popular and developed market segment of Cloud Computing. IaaS deliver huge amount of infrastructure (Processor, Network, Storage) on demand and at pay-as use basis [6]. Hardware virtualization is used to maintain and deliver this kind of service to Cloud user. Hardware Virtualization provides many key benefits such as workload partition, application isolation and encapsulation [15]. From the infrastructure provider point of view, it allows better use to IT resources, and provides more secure way to lease free resources. From the perspective of Cloud user, it reduces the administration and maintenance cost as well as capital cost allocated to purchase of hardware [8].
- Platform as a Service- Building and running on-premise applications has always been complex, expensive, and slow. Each application required hardware, an operating system, a database, middleware, Web servers, and other software [6]. Platform as a Service abstract the infrastructure and support a set of applications program. It is middle bridge between hardware and application. PaaS provide application a runtime environment and do not expose infrastructure to the Cloud user. Deployment process of application to infrastructure is automated by configuring application component, provisioning and configuring supporting technologies such as load balancers and databases [8]. From developer point of view, they design there system in terms of application and are not concerned with hardware (physical or virtual), operating system and other low-level services [16]. Figure 1.3 shows life cycle of PaaS application.

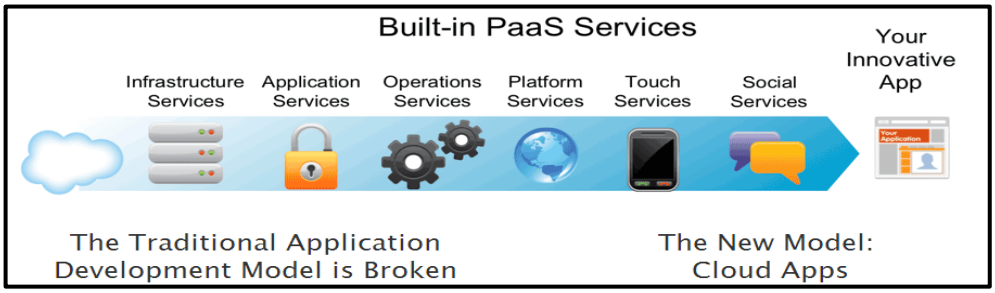

Figure 1.3: PaaS LifeCycle [16]

- Software as a Service- Software-as-a-Service aims at replacing the applications running on PC. There is no need to install and run the special

software on your computer if you use the SaaS. Instead of buying the software at a relative higher price, you just follow the pay-per-use pattern which can reduce you total cost. In this scenario, customer neither need install anything on their premises nor have to pay considerable upfront cost to purchase software and the required licences. They simply access the application website; enter their login and billing details and use the application instantly.

1.5 Cloud Computing Open Challenges

Despite of best efforts from industry and academic research Cloud Computing faces many challenges for its adoption, as shown in Figure 1.4.

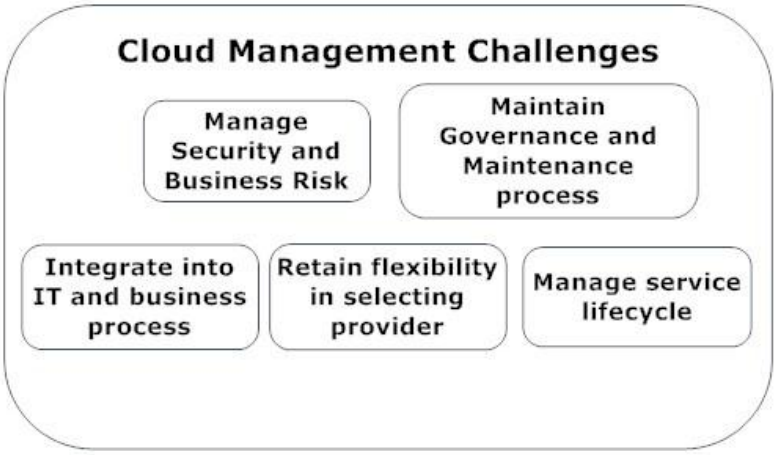

Figure 1.4: Cloud Computing Challenges

Major challenges presented in many papers and websites are discussed here [17] [18] [19]:

- Cloud Definition- As discussed earlier there have been several attempts made in defining Cloud Computing and in providing classification of all the services and technologies similar to Cloud Computing. One of the most comprehensive researches for Cloud definition is done by NIST [20]. They submitted a technical report on defining Cloud Computing. Some other definitions are explained in Section 1.2. These definitions reflect what is meant by Cloud Computing at present stage. As cloud is in very earlier stages and it is constantly evolving so real nature of Cloud Computing will not be capture by these definitions. Definition of Cloud Computing will be an open challenge until Cloud doesn't evolve completely.

- Cloud Interoperability, Portability and Standardization- In order to realize full potential of Cloud Computing introduction of standards and allowing interoperability between solutions offered by different vendors is of fundamental importance. Vendor lock-in is major hurdle in seamless adoption of Cloud Computing. Many Medium and Small Scale Organizations hesitate to adopt Cloud because for lock-in and monopoly of one vendor. Vendor lock-in prevent these organizations from switching to other competitors solution or when it is possible it happens at considerable conversion costs and require significant amount of time. The presence of standards, which are actually implemented and adopted within Cloud Computing community, could give room for interoperability and lessen the risk resulting from vendor lock-in.
- Scalability and Fault Tolerance- Most attractive feature of Cloud Computing is on demand scalability and fault tolerance. Cloud allows the scaling beyond the in-house IT resources whether they are infrastructure or application services. In order to implement these capabilities Cloud should be designed by taking scalability in mind along with performance, size and load. Within this scenario, the ability to tolerate failure becomes fundamental, sometimes even more important than providing extremely efficient and optimized systems. Load on Cloud services are increasing rapidly from last 3-4 years. At present running scalability algorithms will be out dated soon. Hence, the challenge in this area, is designing high scalable and fault tolerance systems, which are easy to manage.
- Security, Trust and Privacy- Security, Trust and privacy issues are major obstacles for massive adoption of Cloud Computing. Data is power in these days. As all data in Cloud Computing is at provider, so user hesitates to give all data to third party. While the data is in memory, it is in decrypt form, but since application is hosted on virtualized infrastructure. It became accessible to all virtual machines hosted in that infrastructure. In this case, there is lack of control over the environment in which application is executed.

  On the other side, user has to decide whether to trust or not trust the provider itself. Many agreements are signed before using Cloud services and proper compensation is provided for violations. But when a violation of privacy and

illegal access to sensitive information is detected, it could become difficult to identify who is liable for such violations in virtualized environment. The challenges in this area are then mostly concerned in devising secure and trustable system from different perspectives: technical, social and legal.

1.6 Research Motivation

Despite the advancement in Cloud Computing from industry and academics contributions, Cloud faces many challenges for its worldwide adoption. The primary reason for not adopting the Cloud is that current cloud solutions are not built considering interoperability and portability. This locks Cloud user to single infrastructure, platform or software provider preventing the portability of user data and software [21]. Likewise big companies like Microsoft, Google, and Amazon are reluctant to agree on widely accepted standards promoting their own standards making it more difficult and complicated. Dominance of big companies increases the lock-in effect and it affects small scale and middle scale companies to enter into the cloud market.

Interoperability and Portability are missing elements that can solve lock-in issues. In portable Cloud environment users can compare and choose from different service providers and can easily switch between Cloud providers without unsettling user data and configuration [22]. A portable Cloud market will open opportunity for small scale and middle scale organizations and strengthen their market place. They will able to cooperate with large organizations to create new business models as per demand without conflicting due to portability. Hence addressing the issue of interoperability and portability is both timely and necessary. However, different packaging standards and framework can possibly lead to different portability solutions which are not compatible with each other.

Semantic linking between applications and data is missing part to connect multiple Cloud services and solve interoperability problems [22]. Semantics of data, action taken on the data and vocabulary in which these actions are expressed, constitutes basics of interoperable Cloud. This thesis work aims to use existing standards and create an architecture using semantics so that cloud user can easily switch to other Cloud providers without any obstructions and thus solve portability issues in PaaS.

1.7 Organization of Thesis

Rest of this thesis is organized as follows:

Chapter 2 – This Chapter summarizes literature survey done to study the concept of Portability, Virtualization, Resource Discovery and Service Level Agreement (SLA) in Cloud Computing.

Chapter 3 – This chapter focuses on research gaps and also state problem statement for this thesis.

Chapter 4 – This chapter provides solution to problem stated in previous chapter by proposing architecture and explaining its all components. Designing of system using UML diagram is also presented.

Chapter 5 – This chapter demonstrates the proposed architecture by implementing it using Protégé 4.0, Vmware Studio, Vmware Workstation etc. tools. Results of experimental setup are shown.

Chapter 6 – This chapter describes the conclusion, contribution of work and future research possible.

Chapter 2

Literature Review

In the previous Chapter, Cloud Computing is introduced by discussing its service layers and chief characteristics followed by some of significant application areas and open challenges. One of crucial open challenges in field of Cloud Computing is Portability. Focusing on stated challenge this chapter discusses state of art and research work in the field of portability of Cloud Computing. This literature reviews and explores four domains decisive for portability in Cloud computing: Portability in Cloud, Standardization in Cloud, Resource Discovery in Cloud, and Service Level Agreement. Structure of this literature review is, firstly domain in discussion is explained in detail then different techniques used in that domain and lastly their relation with Cloud Computing and work done by different researchers in respective domains. Conclusion to this chapter is provided at the end.

2.1 Portability in Cloud

In this domain of literature survey, portability issues and its importance is discussed.

In Cloud Computing model data is transmitted to service provider along with all necessary configurations. This data is remotely accessed by user via internet and update data as required. After several updations over the time, local data present with consumer is out dated. In a scenario where consumer wants to shift data locally or migrate to another Cloud provider and stop using Cloud services before completion of contract. In that scenario, consumer has to pay large amount of termination fee to providers to get their data back [23]. When small scale or middle scale organization realizes that there is risk associated with Cloud Computing, they will be less likely to adopt Cloud Computing. This risk is known as Portability issue in Cloud Computing. Pressman [24] defines portability as the ease with which the software can be transported from one environment to another without any data or configuration lost.

Shirazi et al. [25] defines portability in Cloud as ability to move data applications from one Cloud provider to another without any loss. They provide some reasons why user wants to change Cloud provider. Some of them are dissatisfied from service,

require better alternative options, change in business and technical strategy. Contract plays an important role when user migrates from one provider to another because it mentions the argument conditions. Robert et al [23] proposed to have a data-hosting clause along with arbitration and litigation clauses. Data-hosting clause helps consumers and providers when user wants to move data. It neither allows provider nor user to pay high compensation for migration.

Many new providers enter into Cloud marketplace providing new features and capabilities at lower cost comparing to existing ones. Non-availability of portability prevents users to shift to new users. Many standard development organizations and workgroups are working for achieveing portability in Cloud environment. NTT submited a technical report [26] which describes standardization activities done by different organizations for achieving portability and interoperability. Among different standards, DMTF Open Virtualization Format (OVF) [27] which introduced in 2007, is a first step towards hypervisor independence thus achieving Cloud portability. OVF provides a way to move the virtual machine in form of virtual appliance from one hosted environment to another. OVF standardizes the use of a container that stores metadata of virtual machine and enables the migration of virtual machine. OVF is used as basis in this thesis for achieving portability. Next section explains virtual appliances in detail which are virtual machines ready to run and packaged in OVF format.

2.1.1 Virtual Appliance

Virtualization is continuously changing IT industry and the way people use resources. Most of computer hardware developed today is for the use of single operating system. Virtualization breaks that hurdle and makes possible to run multiple operating system on single computer hardware. To ease software development and reduce external dependencies on other services, such as operating system, Independent Software Vendors (ISVs) can create a single pre-configured pre-installed package that contains application and necessary component required for running that application [28]. This package is known as Virtual Appliance. Like software appliances, virtual appliances are intended to eliminate the installation, configuration and maintenance costs associated with running complex stacks of software. Virtual appliances are usually built on a standard operating system (OS) and run as a virtual machine (VM).

With virtual appliances, ISVs can create a single application stack, reducing the cost and complexity of deployment and management. It is possible to ship preinstalled, preconfigured solutions that allow customers to plug solutions into their computing environments. For customers, deploying and managing software is easier when the applications are delivered in a virtual appliance. The following items summarize the benefits of virtual appliances [29]:

- Accelerate time to market – Customers can quickly download and power-on your virtual appliance.
- Reduce distribution overhead – The same virtual appliance runs on most VMware product platforms.
- Increase reliability – VMware Studio builds an optional update repository for automatic patching.
- Enhance security – Appliances are less vulnerable to security breaches than a general-purpose OS.
- Lower support costs – Virtual appliances require little configuration and no maintenance.

Figure 2.1 shows how a virtual machine is converted into a pre-installed, pre-configured OS and Application stack. The idea has been initially presented [30] to address the complexities of system administration by making labour of applying software update independent of number of computers on which the software is run. Overall, the work develops the concept of virtual network of virtual appliances as a means to reduce the cost of deploying and maintaining software.

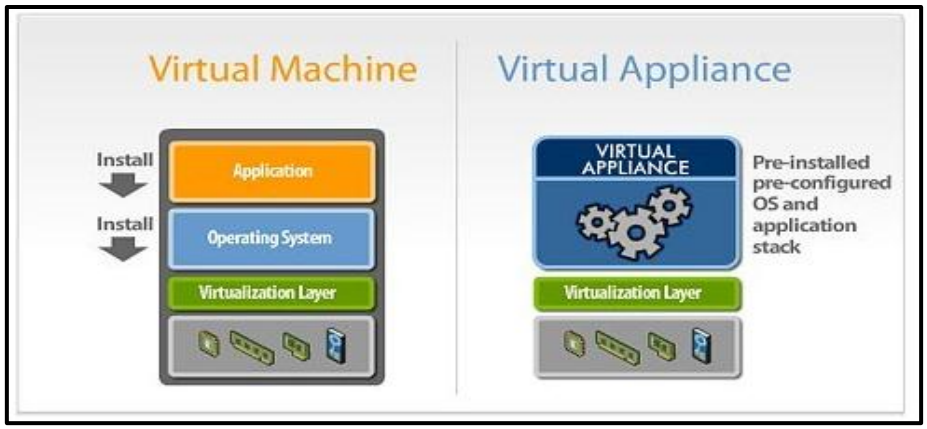

Figure 2.1: Virtual Appliance Stack Configuration

Figure 2.2 illustrates the workflow to create, distribute, and update a virtual appliance. A software vendor packages the software for distribution to corporate customers and end-users, which deploy the package on a virtualization platform. VMware Studio can optionally build an update repository, and embed the URL of the update repository into the virtual appliance, which periodically checks for updates after deployment

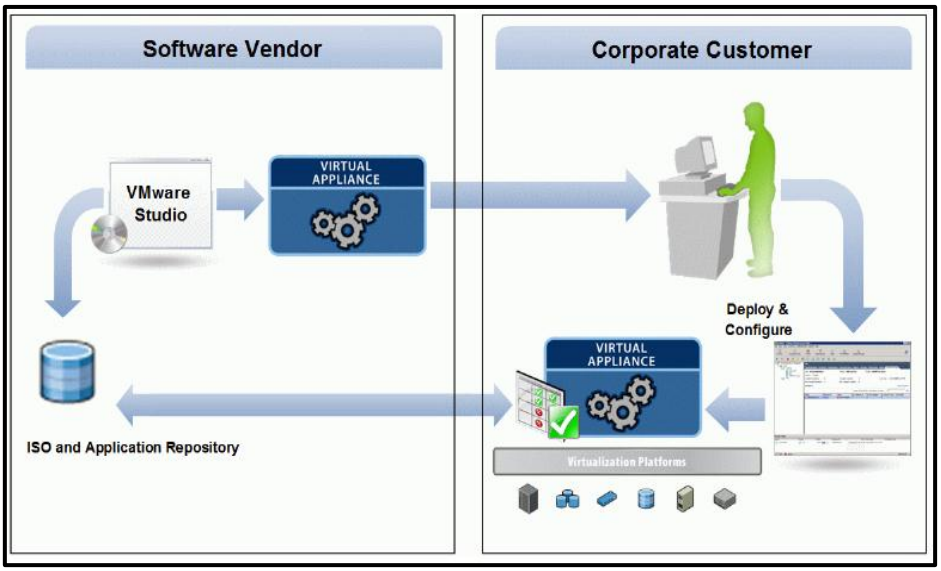

Figure 2.2: Virtual Appliance Workflow [29]

These virtual appliances are packaged in a universally adopted, hypervisor-neutral format known as Open Virtualization Format (OVF). The Open Virtualization Format (OVF) [27] developed by DMTF adopted by ISO and ANSI [31] is a hypervisor-neutral (the OVF doesn't rely on the use of specific hypervisor or virtualization platform), and open specification for the packaging and distribution of virtual appliances composed of one or more VMs. It aims to facilitate the automated, secure management of not only virtual machines but the appliance as a functional unit. OVF is virtualization platform neutral, while also enabling platform-specific enhancements to be captured. This makes it a proper format for Cloud computing where users have to deal with diversity of virtualization platforms.

2.1.2 State-of-the-art in Portability Solutions

Some of the current practices of portability in Cloud Computing are devised in this section.

- Govindarajan and Lakshmanan [32] report that besides APIs and brokers, interoperability should be investigated trough control, data and other

additional issues, such as policy management, security management and deployment /provisioning aspects. Moreover, they propose to build relevant layers of abstraction to help interoperability and portability.

- Mahdi et al. [25] presented a process to achieve data portability among Cloud providers. Design patterns and graphs are used to migrate the databases. However there are many limitations as they don't store foreign keys and are limited to family of design patterns.
- Hill et al. [33] propose to have an abstraction layer between Cloud provider and user. Applications will be developed based on this abstraction layer. This abstraction layer will provide complete and comprehensive method to develop applications which can be easily portable to multiple providers. Moreover due to metadata added with data for recognition by multiple providers makes it complex and time consuming.
- Fermín et al. [34] focuses on how a complex enterprise-class transactional applications can be made deployment-agnostic by means of the parameterization mechanisms offered by the Open Virtualization Format (OVF) standardized by the Distributed Management Task Force (DMTF), and then customized at deployment time (either automatically or by requesting user input) by means of the OVF activation mechanism. However they tested their proposed solution in Vmware Hypervisors. Portability issues across different hypervisors are still to discuss.
- Dastjerdi et al [35] enlighten the use of virtual appliance and OVF as a tool for getting portability in Cloud environment. They proposed an architecture which shows deployment of infrastructure provider as a virtual appliance. But their architecture is limited to IaaS only and no support for SLA and QoS is provided.

This section discusses portability in Cloud Computing. Next section describes result of a survey conducted on standardization initiatives in Cloud Computing.

2.2 Existing Standardization Initiatives in Cloud Computing

Standardization can be defined as [20] a process of developing standards so that independence on single provider can be removed. Standardization enables many non-functional properties such as compatibility, portability, interoperability, safety and

quality. Cloud Computing lacks in universally adopted and followed standards. Standard should be generic, high level conceptual model which should be powerful tool for discussing requirements, use cases, frameworks and operations in Cloud Computing [36]. As discussed in an article by Sixto Ortiz [37] many industry experts speak about the importance of standards in Cloud Computing. Dan Kusnetzky, IT analyst of The Kusnetzky Group, discuss the crucial importance of standardization in early stages of Cloud Computing and raised the question of incompatibilities that arise due to lack of standardization in Cloud Computing. Lynda Stadtmueller, program director for Cloud Computing at Frost & Sullivan's Stratecast, mention that the lack of standardization is causing difficulties for buyers to compare and evaluate Cloud offerings. Nirlay Kundu, Senior manager at Wipro Consulting Services, talk about security standards by addressing issues such as data privacy and encryption which is hurting wider Cloud-Computing adoption in recent times. Winston Bumpus, Director of Standards Architecture at VMware, acknowledges the development of next set of standards which will revolutionize compute and data sharing in Cloud Computing.

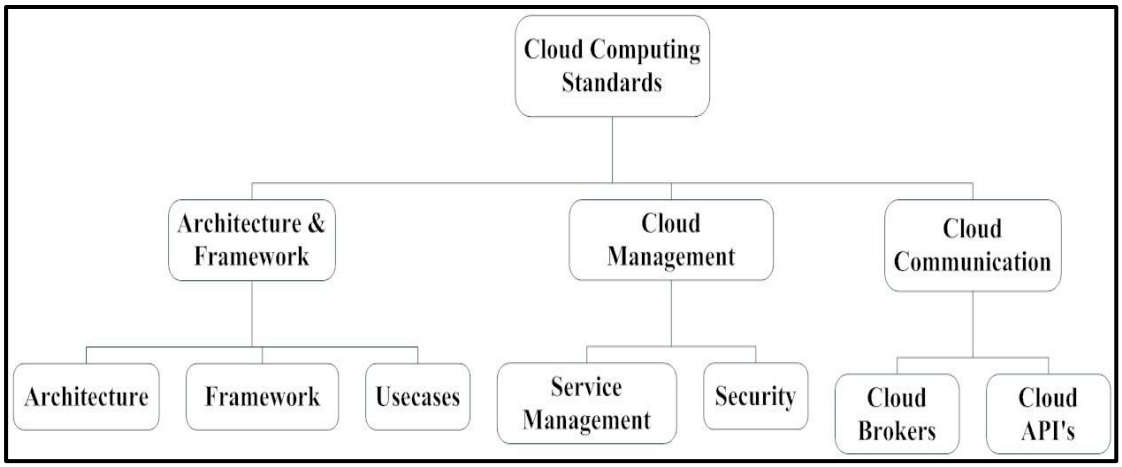

Figure 2.3: Categorization of Standards

As debated by many industry experts, standardization is very important for Cloud Compuitng. To recognize current state in standardization for Cloud Computing a systematic and comprehensive survey is conducted for finding current standard initiatives by different Standard Development Organizations (SDO), Technical Forums and Government Organizations. Survey facilitates to understand different standards, their standardization area, and gaps in standardization of Cloud Computing. From the survey 31 Standard initiatives from 20 different organizations were identified. These Cloud standard initiatives were categorized according to working

area of different workgroups into three categories viz., Architecture & Framework, Cloud Management and Cloud Communication, as shown in Figure 2.3 above.

Of the three main categories shown in Figure 2.3, 10 standard initiatives were found in Architecture & Framework and Cloud Management and 11 standard initiatives were found in Cloud Communication, as shown in Figure 2.4. There is much progress in Cloud standardization fields from last 5 years. Figure 2.5 shows the status of all standards on the basis of the 6 stages described by NIST [36]. It depicts two standards were adopted by Standard Agencies, four standards are in the implementation phase and rest are developing profiles for their respective standards. Major milestone was adoption of OVF by ISO and ANSI as standard for packaging of virtual machines [31]. Other significant standard initiatives by OCCI, CSA, and SNIA are getting world-wide acceptance.

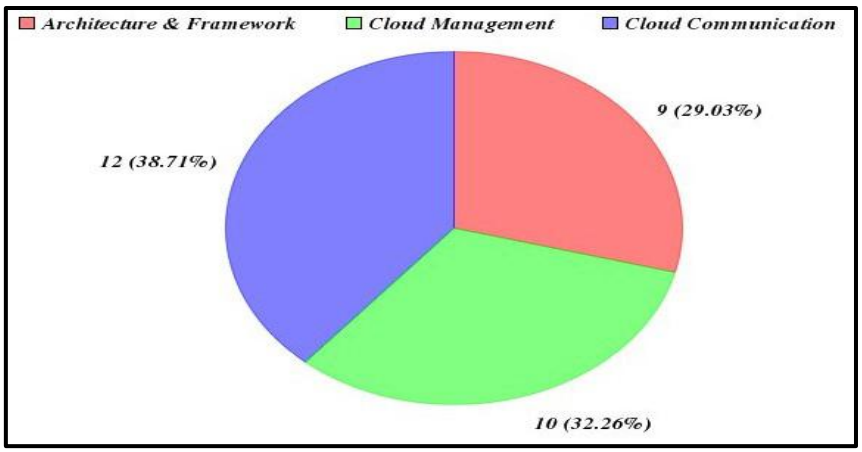

Figure 2.4: Contribution of Groups in Discovered Cloud Standards

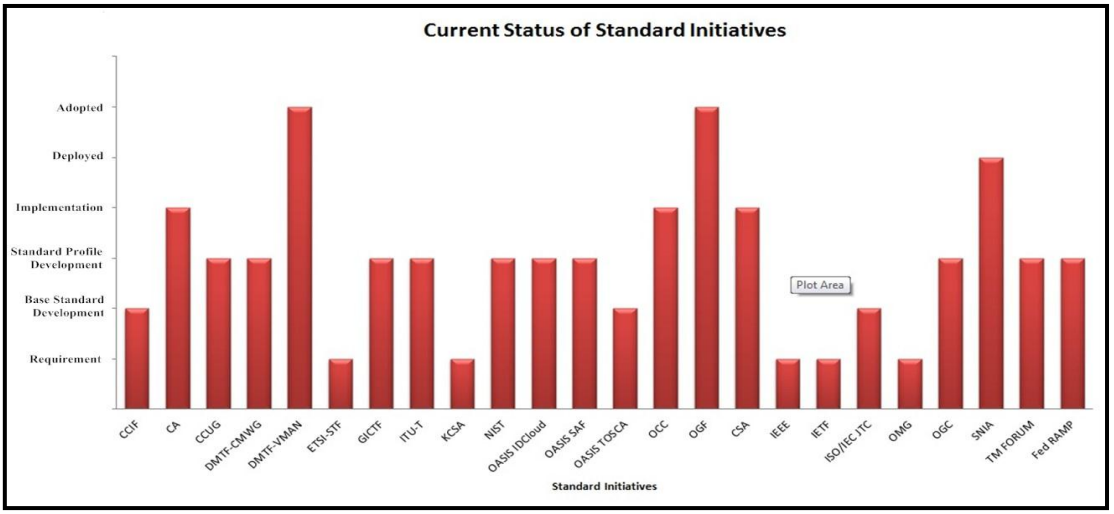

Figure 2.5 Current Status of Each Standard Initiative

Some of important standards and standard bodies are:

I. *National Institute of Standards and Technology (NIST)*: NIST is a technical department belonging to the U.S. department of commerce. NIST aims to shorten the adoption cycle, which will enable near-term cost savings and increased ability to quickly create and deploy enterprise applications [38].

II. *Open Cloud Computing Interface (OCCI):* Open Grid Forum (OGF) has announced Open Cloud Computing Interface(OCCI) in 2009 and released API specifications for Infrastructure as a Service (IAAS) [26]. It supplies a general purpose set of specifications for Cloud based interaction with resources.

III. *Distributed Management Task Force- Virtual Management (DMTF-VMAN):* DMTF VMAN is a standard which includes a set of specification that addresses the management Lifecycle of virtual environment. They developed a standard format for packaging virtual machines known as Open Virtualization Format(OVF) which helps to make virtual machines portable [27]. It is a packaging standard not runtime standard and it is universally adopted. Many tools are developed which uses this standard for secure and efficient packaging of virtual machines.

IV. *Storage Network Industry Association (SNIA)*: SNIA has formulated the specifications of Cloud Data Management Interface (CDMI) which is an API for controlling storage units. It provides storage for Cloud applications in elastic, on-demand and bill as you use basis [39].

Some of standards have been explained above. Rest of the standards [26] are listed chronologically in Table 2.1. Table 2.1 also states contribution which each standard initiative is indented to provide after its completion and adoption.

Table 2.1 Cloud Computing Standards Initiatives

Category	Standard Name	Year	Contribution of Standard
Architecture & Framework	CSA	2008	It studies best practices in Cloud security and governess.
	KCSA	2008	The main activities of KCSA are the creation and promotion of Cloud services.
	CCUG	2009	Its goal is to bring Cloud consumer and Cloud vendor to define

			common use cases for Cloud implementation.
	ISO/IEC JTC	2009	It provides Definition, Terminology, Framework for Cloud Computing
	OCC	2009	It primarily focuses on managing and operating Cloud infrastructure to support scientific, environmental and healthcare research.
	OGC-WG	2009	This working group is responsible for creating a common understanding between buyers and suppliers of Cloud Computing.
	OMG	2009	It is responsible for making coordination among various standard bodies and groups.
	ITU-T	2010	This group works in standardization of Cloud from the telecommunication perspective.
	NIST	2010	Architecture, Definition, Use cases
Cloud Management	DMTF-VMAN	2007	Virtualization and Resource management in Cloud Computing
	CSA	2008	Security and Risk Management for Cloud Environment
	OGF	2009	Scaling, Monitoring, Deployment of Cloud Services
	TM Forum	2009	It started Enterprise Cloud Leadership Council (ECLC) in 2009 to resolve issues in standardization, security, performance etc.
	CA	2010	It provides a common interface and namespace that allow Cloud providers to automate the audit and assessment process of their environment and allow authorized consumers to do likewise via an open, extensible and secure API's
	IETF	2010	Resource Management
	ITU-T	2010	Security
	NIST	2010	Security, Monitoring
	OASIS IDcloud	2010	Identity deployment, provisioning, monitoring
	Fed RAMP	2012	Assessment, authorization, monitoring
Cloud Communication	ETSI-STF	2006	This group works in grid Computing, IT to telecom convergence and in particular, the lack of interoperable grid solutions
	DMTF-VMAN	2007	It tries to solve portability issue of virtual machines by packaging them in universally adopted standards.

	CCIF	2009	It works for development of an ontology framework in which one or more organization can work together.
	DMTF-Incubator	2009	It develop a Cloud Standard incubator in 2009, which provides use cases and reference architecture to build an interface between Cloud service provider and consumer.
	GICTF	2009	GICTF deals with creation of usecases, requirements and specifications for inter-Cloud communication.
	OASIS SAF	2009	OASIS started this Technical committee (TC) in 2009 to automate appropriate response to changing business conditions and integrate it to all domains
	OGF	2009	Open Grid Forum (OGF) has announced Open Cloud Computing Interface(OCCI) in 2009 and released API specifications for Infrastructure as a Service (IAAS).
	SNIA	2009	API's for storage
	ITU-T	2010	Portability and interoperability
	NIST	2010	API's for IAAS
	IEEE	2011	Portability, Interoperability standard development
	TOSCA	2012	Interoperability and portability

This section summarizes all standard development initiatives in Cloud Computing. Major outcome form this survey was adoption of OVF universally. Further literature survey is done keeping OVF in mind and how portability can be achieved using this packaging standard. In next section third important domain for portability i.e. Cloud Resource Discovery is discussed.

2.3 Cloud Resource Discovery

Resource discovery in any environment whether it is Grid Computing or Cloud Computing is very crucial task. If one can easily discover resource providers according to its requirement at first time, need of portability reduce to significant amount [40]. Main objective of this domain is to select most capable provider according to user requirements. This selection process is carried out by a Cloud Broker.

Cloud broker is an entity that manages the use, performance and delivery of cloud services and negotiates relationships between Cloud providers and Cloud consumers. A simple example of cloud broker – as an intermediation service – is the price calculation service provided by many cloud vendors, e.g. monthly calculator by Amazon Web Services. It is challenging to broker services across multiple vendors since the cloud service specifications are non-standardized and thus can be highly heterogeneous and pose semantic interoperability issues [41]. With increase in diversity of Cloud brokers in market, resource discovery is very complex and crucial task. Functional as well as non-functional requirements should match. Raman et al. [42] states that exact terms matching systems used now-a-days bears many limitations and Cloud ontology with proper semantically matching algorithm can overcome these limitations. Next section explains ontology and its language in detail.

2.3.1 Ontology and OWL

Ontology describes concepts in a specified domain and also lists relationship between those concepts [43]. In this section Ontology and Owl are explained in detail.

There are many definitions of Ontology:

- *"An ontology is a formal specification of a shared conceptualization."* [44]

- *"The main thread of ontology in the philosophical sense is the study of entities and their relations. The question ontology asks is: What kinds of things exist or can exist in the world, and what manner of relations can those things have to each other? Ontology is less concerned with what is than with what is possible."* [45]

From above two definitions, ontology can be described as a web of nodes and edges where nodes represent concepts and edges as relationship between these concepts. To create ontology an ontology language is required. There are many ontology languages but most common used are OWL and RDF [44]. OWL is used to develop ontologies in this thesis so OWL is discussed below.

Like all other languages OWL makes it possible to describe concepts but it also provides new facilities. It has a richer set of operators - e.g. intersection, union and negation. It is based on a different logical model which makes it possible for concepts

to be defined as well as described. Complex concepts can therefore be built up in definitions out of simpler concepts. Furthermore, the logical model allows the use of a reasoner which can check whether or not all of the statements and definitions in the ontology are mutually consistent and can also recognise which concepts fit under which definitions [43]. OWL has some basic properties which helps it to represent relationships. There are two main types of properties, Object properties and Datatype properties. Object properties are relationships between two individuals. Object properties link an individual to an individual. OWL also has a third type of property Annotation properties. Annotation properties can be used to add information (metadata | data about data) to classes, individuals and object/datatype properties [43]. Figure 2.6 depicts example of all three types of properties.

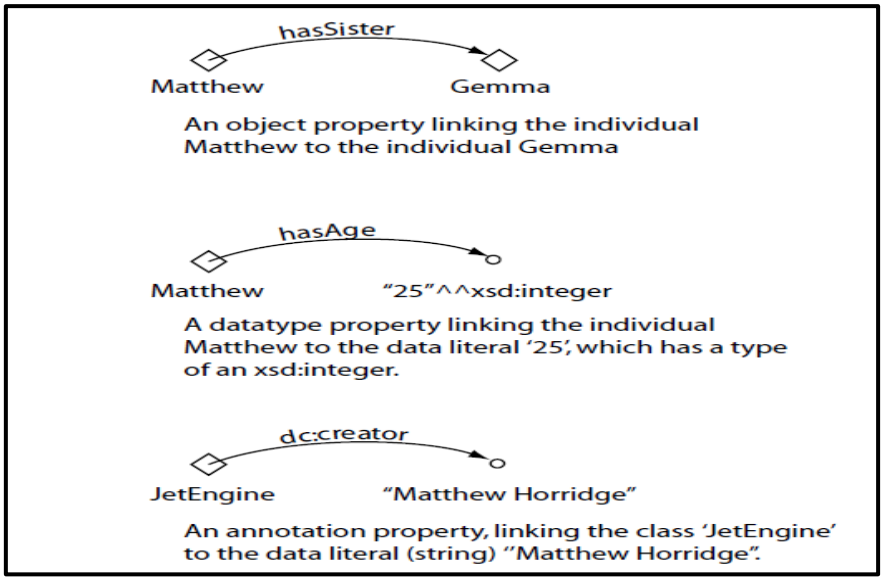

Figure 2.6: OWL Properties [43]

As explained above ontology is a web with nodes and edges. Nodes in Cloud ontology web are providers and edges are the requirements. Major hurdle after creation of Cloud ontology web is finding similarity between nodes in this case providers. Knappe et al. [46] presents a method which finds similarity between two nodes in a Graph known as Similarity Graph Algorithm. This algorithm is used in this thesis for finding similarity between two providers and it is explained in next section.

2.3.2 Ontology Similarity Matching

The objective of this section is to devise and discuss principles for evaluation of similarity measures that utilizes knowledge from ontology to obtain better and closer

answers on a semantically level, thus comparing the concepts rather than exact words. Ontology stays at back end after defining concepts and relationship between these concepts. In combination with a given basic ontology, a concept language, also known as description language, is used which defines a set of semantic relations which can be used for formation of compound or mixed concepts [46]. The suitable number of available relations may vary with different domains, but among the more general relations that probably will be present in most domain modelling are WRT (With-respect-to), CHR (Characterized-by), CBY (Caused-by), TMP (Temporal), and LOC (Location). Take as an example the sentence: "*the black dog is making no*ise" which can be translated into this semantic expression *noise[*cby*: dog[*chr*: black]]*. This semantic expression says *noise* Caused-by *dog* Characterized-by *black.* So, English line is converted into a semantic expression which can be easily applied to create a similarity graph shown in Figure 2.7 [46]. In the same way, a semantic expression of user requirements is created using ontology. Semantic Graphs are explained below.

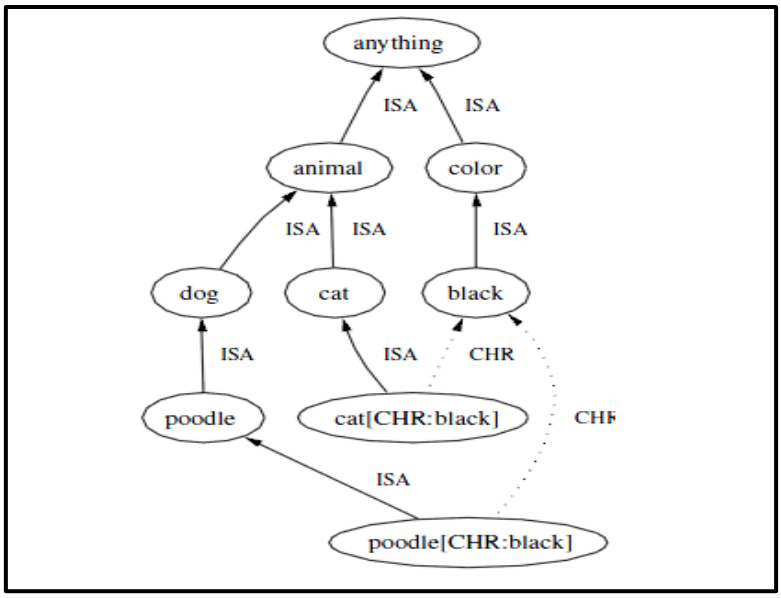

Figure 2.7: Sematic Graph of Semantic Expression [46]

A similarity graph is a subpart of the ontology represented as a graph with a subset of concepts as nodes and relations connecting these as edges. Similarity graphs can be defined for any set of one or more concepts and specifically use the notion as a basis for similarity based on graph computations. The similarity between two concepts can thus be derived from a similarity graph covering these concepts.

Matching of concepts is done by using Similarity Graph algorithm [46]. According to Similarity Graph Algorithm similarity between two nodes in a similarity graph can be determined as:

$$sim(x, y): C \times C \rightarrow [0,1] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\text{ I })$$

Where C is well formed concepts and $sim(x, y)$ measures the degree to which y is similar to x. Extreme values $sim(x, y) = 0$ means not similar and $sim(x, y) = 1$ means fully similar. The latter will only be case when x = y.

Andreasen et al. [47] suggested that to find similarity between two concepts (nodes) in a similarity graph just calculate shortest distance between two concepts (nodes). Two nodes with shortest distance will be similar. But this method is very complex and not suitable for large query systems. Knappe et al. states $\propto (x)$ be nodes from x to uppermost node and $\propto (y)$ be nodes from y to uppermost. So similarity between x and y can be determine with number of nodes common i.e. $\propto (x) \cap \propto (y)$. Similarity factor can be written as:

$$sim(x, y) = \frac{\propto(x) \cap \propto(y)}{\propto(x)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\text{ II })$$

Similarity between two concepts can be find using above similarity factor. Two concepts having highest similarity factor are most similar ones. Next section enlist some of work done by researchers for resource discovery in Cloud Computing using semantics.

2.3.3 Semantic Resource Discovery in Cloud

Based on the technologies explained above many cloud researchers develop semantic search engines for discovering resources in Cloud environment. Some of them are :

- Kang et al. [48] developed a search engine, called Cloudle, for discovering cloud services under an agent-based paradigm for Cloud resource management. Cloudle adopts an ontology-based strategy for service discovery. The objective is to return pages containing relevant services as in standard search engines.
- Dastjerdi et al [35] investigated cloud service discovery at an architectural level and in a much broader context of appliance management on cloud. The approach offers a mechanism for users to choose IaaS providers and facilities

the way to deploy the services. The architecture comprises a semantic matchmaker based on WSMO framework.

- Godse et al [49] presents an approach to select a service by using Analytic Hierarchy Process (AHP) technique for prioritizing the service features.
- Ding et al. [40] proposed a hybrid technique based on syntactic and semantic of input and output of the services for cloud service discovery. However, this approach is based on IO Matching alone.
- Jaeyong et al. [50] proposed a multi-agent architecture for service discovery in Cloud environment by using Ontology matching. To increase the utility and success rate of matching consumers' requests to resources, they use a database to store and keep track of historical data for making intelligent recommendation based on attribute value prediction. This architecture however doesn't consider any QoS or actual deployment parameters.

2.4 Service Level Agreements (SLA)

Quality plays an important role while delivering services in Cloud Computing. There are many providers fulfilling functional requirements but very less meet the quality requirement. So, there is need of Service Level Agreement (SLA). SLA can be defined as a contract between the service consumer and service provider that defines that level of service. It is SLA that defines the quality of delivered service and ensures right information gets to the right person at right time, safely and securely. Service Level Agreement provides a set of specification by which quality of delivered service can be verified and actions can be done if it is not achieved [51]. One thing to note is SLA cannot guarantee that all promises will be kept and it cannot make a good service out of bad one, but it defines what will happen if stated promises are not kept. A proper SLA defines following things [51]:

- How delivery of the service at the specified level of quality will become realized?
- Which metrics will be collected?
- Who will collect the metrics and how?

- Actions to be taken when the service is not delivered at the specified level of quality.
- Penalties for failure to deliver the service at the specified level of quality.

Most important feature of Cloud Computing is pay-as-use which totally depends on quality of service. No one will use Cloud services if they are taking more time, money than conventional methods due to bad quality of service. If quality of service is pre-discussed and signed then everyone is bond to provide that level of service which increases adoption and trust of consumers in Cloud Computing. It is common for IT service providers to deliver services at different level of quality based on different price range. A SLA is very valuable for helping all parties understand cost, schedule, and performance because their relationship is stated explicitly. Importance of SLA is because it tells two parties about who is responsible for what, what each party will do, and sometimes more importantly what each party will not do [52].

Creation of SLA is very complex and difficult. All parties should be benefitted from SLA. Service provider should know correctly the risk of violating any SLA before signing it. Mastroeni et al. [53] examine the risk of violating the SLA obligations. They evaluate the probability that a SLA commitment on the service availability is violated, when the service restoration time follows an exponential, Weibull, or lognormal distribution. They show that the probability of SLA violation decreases as the variance of restoration times grows, and that lengthening the time interval over which the availability targets are examined is convenient for the service provider just if the compensation amount for each violation grows quite less than proportionally with the length of that time interval.

Further, Zhang et al. [54] discuss varies compensation mechanism in Cloud environment. First mechanism deals with providing compensation whenever any violation is there. Second mechanism works on average compensation and third mechanism states that user and provider should discuss compensation rules before signing SLA.

Moreover, Cloud needs a framework for development and checking for SLA. Brandic et al. [55] represented a layer framework for SLA violation in Cloud environment, they state it as LAYSI. They also present an approach for mapping low-level resource

metrics to SLA parameters necessary for the identification of failure sources. Second, they devise a layered Cloud architecture for the bottom-up propagation of failures to the layer, which can react to sensed SLA violation threats. They also present a communication model for the propagation of SLA violation threats to the appropriate layer of the Cloud infrastructure, which includes negotiators, brokers, and automatic service deplorer.

2.5 Conclusion

In this chapter, state of the art work is discussed related to four different domains of Cloud Computing. Starting with issues and importance of portability in Cloud Computing and Standardization initiatives has been surveyed. The important outcome of both domain reviews is OVF and its adoption universally. After that a literature review is done for Cloud resource discovery using ontology and lastly work in the Service Level Agreement area is reviewed. Based on research gaps found in this literature review next chapter formulates the problem statement.

Chapter 3

Gap Analysis and Problem Statement

Based on literature survey in the previous chapter, gaps have been identified in four domains of Cloud Computing. Gaps identified in this chapter have been recognized by many research communities but proper solution is not known till date. Based on these gaps problem statement for this thesis has been formulated in next section. Objectives of this thesis are enlisted at the end of this chapter.

3.1 Gap Analysis

As in previous chapter, literature survey is done in four domains of Cloud Computing. Based on that, gap analysis is also divided into four domains. Next sub-sections discuss gaps in standardization of Cloud Computing.

3.1.1 Standardization Initiatives in Cloud Computing

Standards are very important in any field of Computer Science. With standards any application build can be adopted universally. Cloud Computing lacks in widely adopted standards. Table 3.1 enlists different gaps in standards of Cloud Computing that need to be developed divided into three categories based on different workgroups.

Table 3.1: Cloud Computing Standardization Gaps

Standardization Area	**Gaps**
Architecture & Framework	I) Standard for Cloud Service Level Agreement (SLA) and Quality of Service (QoS). II) Standard for discovering Cloud services. III) Standard for Cloud security and privacy. IV) Standard for Cloud Reference Architecture.
Cloud Management	I) Standard for Cloud user account and credential management. II) Standard for metering and billing for Cloud service. III) Standard for Cloud identity management.
Cloud Communication	I) Standard for Data and Metadata format. II) Standard for Cloud API Architecture.

3.1.2 Portability in Cloud Computing

Portability is ability to move data or application from one provider to another provider without any loss and large cost. This section identifies some of gaps in portability of Cloud Computing applications, which are:

- There is no portability factor maintained in policy management, security management and data management of Cloud Computing.
- Proper creation of contracts keeping portability in mind. Neither provider nor user should suffer due to portability.
- Risk management due to portability.
- Architecture to facilitate the use of virtual appliance and OVF.
- Common API's for all provider and users.

3.1.3 Cloud Computing Resource Discovery

Resource discovery is crucial for user as well as provider. User can easily get most capable provider whereas small providers can get recognition due to their better deals and facilities. However, this domain needs much work to be done. Some of gaps identified in literature survey done in previous chapter are:

- Most of search engines developed are for IaaS only. PaaS and SaaS search engines need to be developed.
- Need of semantics is encouraged by many research communities but not any significant outcome has come into market.
- Single API for registration of provider and user.
- Proper management of provider and user lifecycle.

3.1.4 Service Level Agreement

Service Level Agreement (SLA) creation and maintenance for Quality of Service (QoS) in Cloud Computing is major hurdle in adoption of Cloud Computing. Some of gaps found in this area are:

- Metrics to be choosen for SLA creation are not standardized. This causes problem when user shifts from one provider to another. Quality is described in different metrics which is very confusing and complex for end user.

- There are few SLA templates for PaaS and SaaS.
- Any method to check statistics provided by provider for quality delivered. Much number of times user feels desired quality is not achieved but provider shows it is achieved. Third party SLA managers are needed to ensure that.
- A portal so that third party SLA manager can advertise about their services.

In this section some of the key gaps have been found during literature survey in four domains of Cloud Computing. Next section discusses the problem statement of this thesis.

3.2 Problem Identification

This thesis addresses four different issues in Cloud Computing in recent times.

I. Data portability in Cloud environment can be defined as ability to reuse or move data between different providers without any change or security issues. Rate at which data is fed into cloud providers is not at which it is out. Data is trapped with single provider so it forces cloud user to stay with one service provider. Achieving data portability is difficult because cloud providers uses different models, programming paradigms and market their own version of same technology. These models are difficult to change or adapt because they are transparent to cloud user. A solution to this problem is storage of data at third party provider.

II. Virtual Machine portability refers to system in which one can reuse virtual machines with different infrastructure provider without any change required and at least achieving same performance. Virtual Machine portability is difficult because infrastructure providers use different hypervisors, provider specific tools and different type of virtualization. Every resource is in virtual form in cloud environment. Achieving portability at resource (infrastructure) level requires secure and complete migration of virtual machines from one infrastructure provider to another. Packaging virtual machine in OVF and its use as virtual appliance is one of the solutions to this problem.

III. Platform and Infrastructure providers have increased in cloud market place in recent times. Proper discovery of these providers by Cloud users is very important. Great deals of limitations are found in current discovery

mechanism because they perform exact match for requirements, due to which it is very difficult to find most capable provider in the market. Current discovery algorithms also do not consider Quality of Service into account while searching. Semantic matching with quality as a factor is need of hour for proper Cloud resource discovery.

IV. Cloud resources are based on Service Level Agreement (SLA) which states usage terms and conditions and proper compensation for violations. QoS information provided by cloud provider can't be trusted because data source is in control of resource provider rather than Cloud user. How Cloud user will know its SLA is achieved or not? Process is required to specify and manage SLA so that information can't be change or false. Proper compensation algorithm is needed when there is any violation.

In this section four different issues in Cloud computing are discussed. A solution to each issue is also provided. Solving these problems in a single architecture is what Cloud Computing needs now. Next sections states objectives of this thesis so that above issues can be resolved.

3.3 Objectives

The objectives of this thesis are:

1. To develop a framework that aims to separate user data, infrastructure provider and platform provider from a PaaS offering.
2. To develop a semantic based searching and match making process for resource discovery.
3. To create a common API using a web-portal for registration of Cloud users, providers and SLA managers.

Chapter 4
Proposed Solution

The gaps and the objectives of this thesis have been explained in previous chapter. In this chapter a solution to automate the management of PaaS environment and resolve some of portability issues have been discussed. Requirement analysis is done using UML designing based on objectives of this thesis. Lastly architecture for automated PaaS management has been proposed. Each component of architecture is explained in detail.

4.1 Requirement Analysis

In this section requirements are collected based on which architecture will be developed. Main focus of the proposed solution is to separate three basic layers of any Platform as a Service provider. These three layers are:

I. **Platform Application Provider**: This provider provides platform that user uses to develop any application or use previously built applications. Examples are Aneka, Visual Studio etc.

II. **Application Data**: In classic PaaS approach user data resides with PaaS provider. In proposed solution, PaaS provider has no control over user data. User data resides at separate IaaS and it is accessed by application at PaaS provider using internet. So, whenever user wants to change PaaS provider he does not have to worry about any data loss.

III. **Infrastructure**: Infrastructure is needed for any PaaS offering. In the proposed solution Infrastructure is separated form Platform provider. Third party infrastructure is proposed to use. This gives total freedom for user to select any infrastructure provider.

When these three layers are separated and linked through network for information exchange many issues can be resolved. For example, user data held by platform provider is secure if migration takes place because data backup is available at another third party infrastructure provider. Some of requirements of this solution are:

I. There are basic 4 actors in this scenario. User, Platform Provider, Infrastructure Provider and SLA manager. Account of all these actor should be managed from a web portal where these actors login or signup. So, first requirement is Login and SignUp of actors.
II. Secondly, provider should able to register their services and user should able to search most capable services. Semantic matching should be used for making matching process more effective.
III. After platform provider is selected then infrastructure provider should be searched. Virtual appliance should be created using OVF packaging format so that it can be hypervisor neutral.
IV. A SLA manager should be assigned to create and maintain QoS.

Above written are basic requirements of proposed solution. To elaborate these broad requirements and get a clear picture UML designing is used in next section.

4.2 Design of the Proposed Solution through UML

The Unified Modelling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of software system. The UML is appropriate for modelling software systems ranging from enterprise information system to distributed web based systems and even to hard real embedded systems. It is very expressive language, addressing all the views needed to develop and then deploy the system. The UML is just a language so it is one part of software development. The UML process is independent, architectural centric, iterative and incremental [56].

In this section, UML diagrams are explained for proposed solution.

4.2.1 Use Case Diagrams:

Use case diagrams are one of the diagrams in the UML for modelling the dynamic aspect of system. Use case diagrams are central to modelling the behaviour of a system, a subsystem, and a class. Each one shows a set of use cases and actors and their relationship [56].

Use case diagrams for our architecture are as follows:

Architecture main use case - This use case, as shown in Figure 4.1, represents the main activities done in this propose architecture. All actors and requirements are shown.

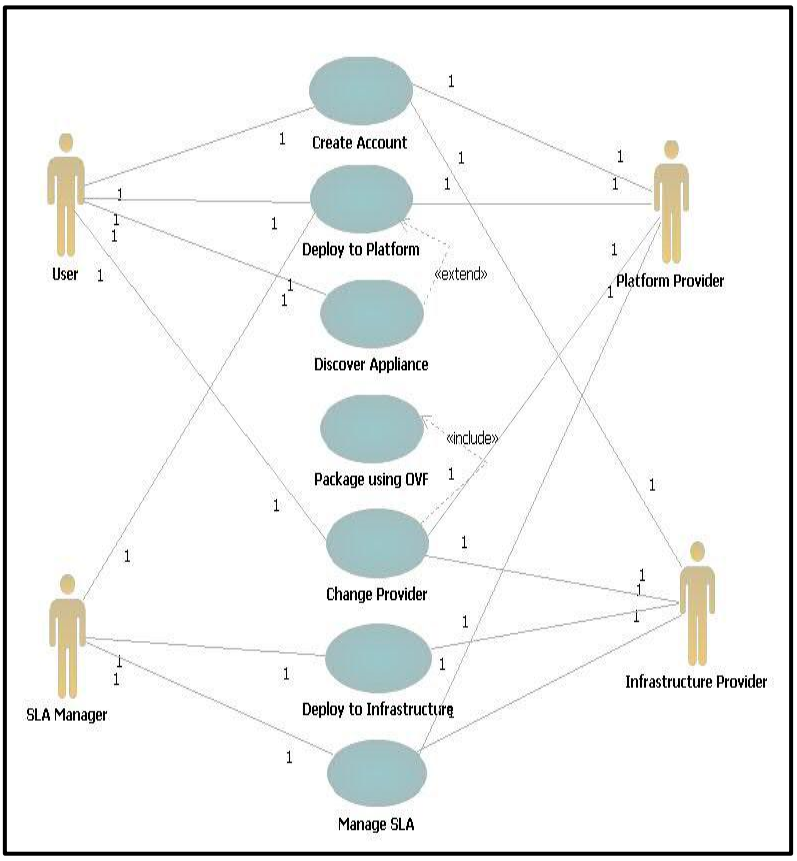

Figure 4.1: Main Use Case of Architecture

Platform Deployment Use Case - As shown in Figure 4.2, this use case depicts main activities related to selection and deployment of platform provider. User can search platform provider, request the provider for using its services. Platform provider accepts or rejects the request. If accepted SLA manager create an SLA between user and provider based on quality of service and store it.

Infrastructure Deployment Use Case - As shown in Figure 4.4, this use case depicts main activities with infrastructure provider in this propose architecture.

Service Level Agreement (SLA) Management Use Case - Figure 4.5 shows use case diagram of Service Level Agreement (SLA) Management activities. All SLA are verified by third party SLA manager and proper compensation is also generated if any violation occurs.

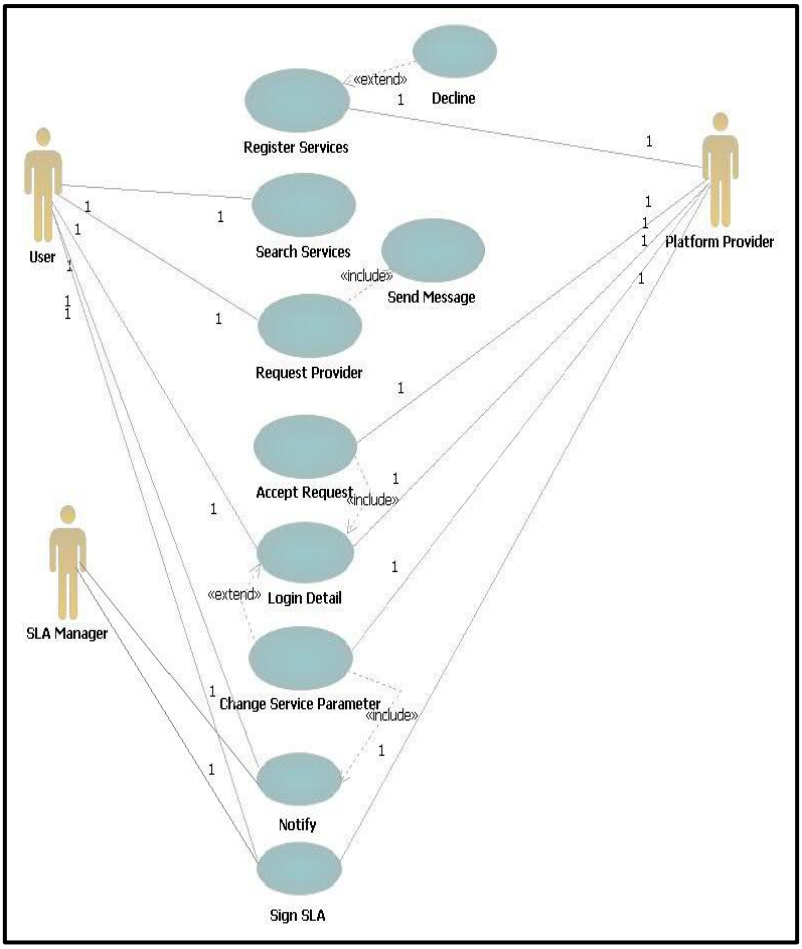

Figure 4.2: Platform Deployment

Appliance Life Cycle Management Use Case - Each virtual machine is converted into an appliance and its life cycle is managed using Open Virtualization Format (OVF) packaging tool. Figure 4.3 shows main activities for appliance management.

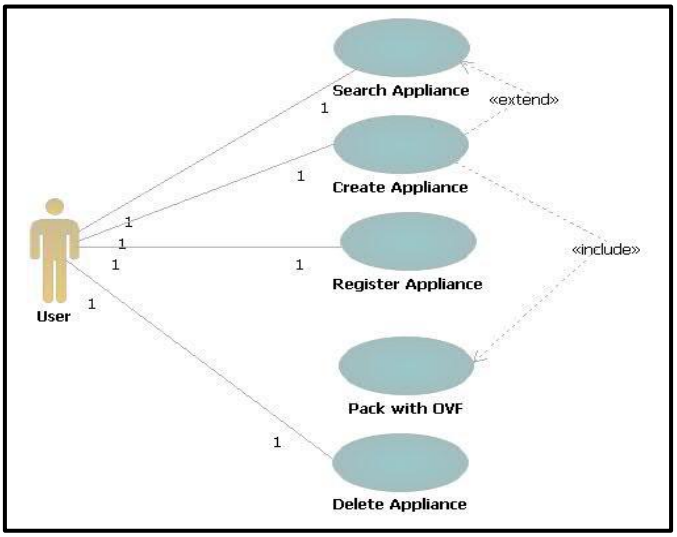

Figure 4.3: Appliance Life Cycle Management

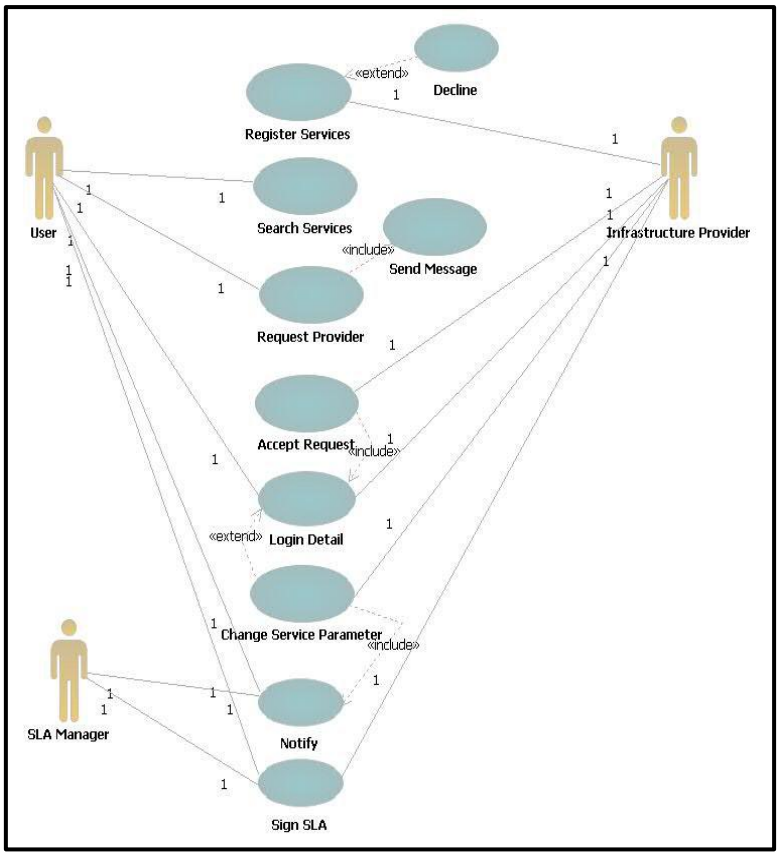

Figure 4.4: Infrastructure Deployment

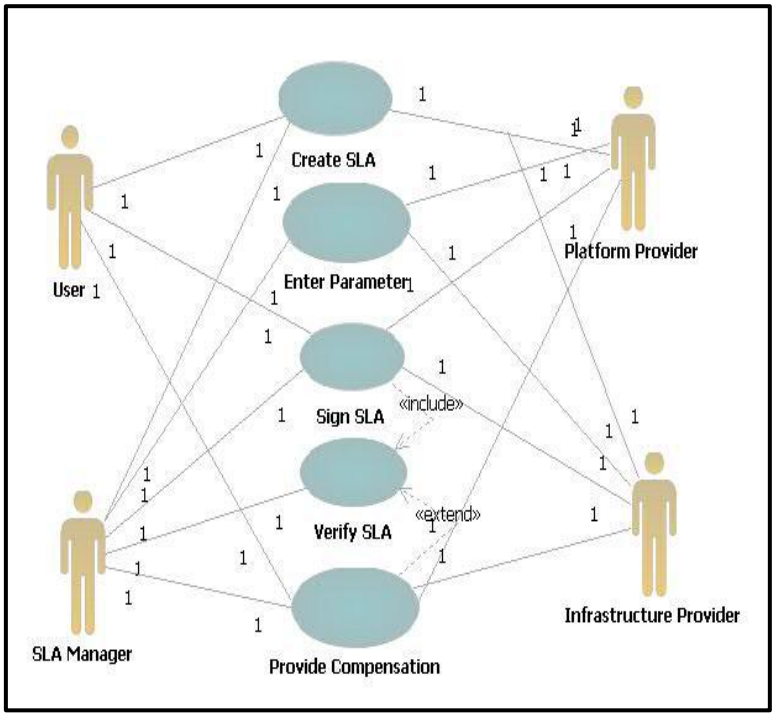

Figure 4.5: Service Level Agreement Management

4.2.2 Activity Diagrams:

An activity diagram is essentially a flow chart, showing flow of control from activity to activity. Unlike traditional flow chart, an activity diagram shows concurrency as well as branches of control. An activity is on-going non-atomic structured execution of behaviour. Actions compass calling other operations, sending a signal, creating or destroying an object, or some pure computation such as evaluating an expression.

This section shows two activity diagrams of proposed solution:

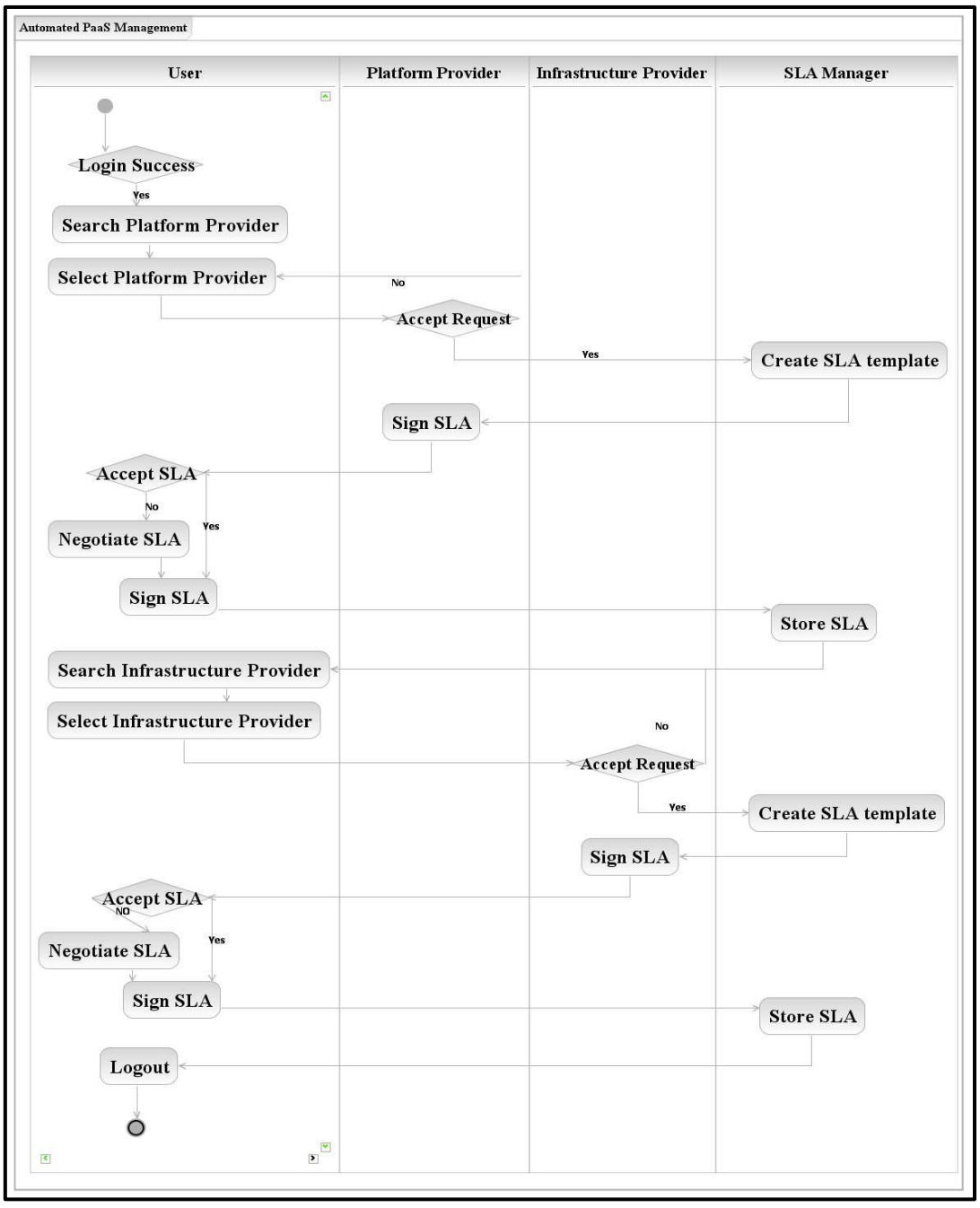

Figure 4.6: Selecting Platform and Infrastructure Provider

Above activity diagrams depicts the activities to select Platform and Infrastructure provider and signing their SLA with SLA manager.

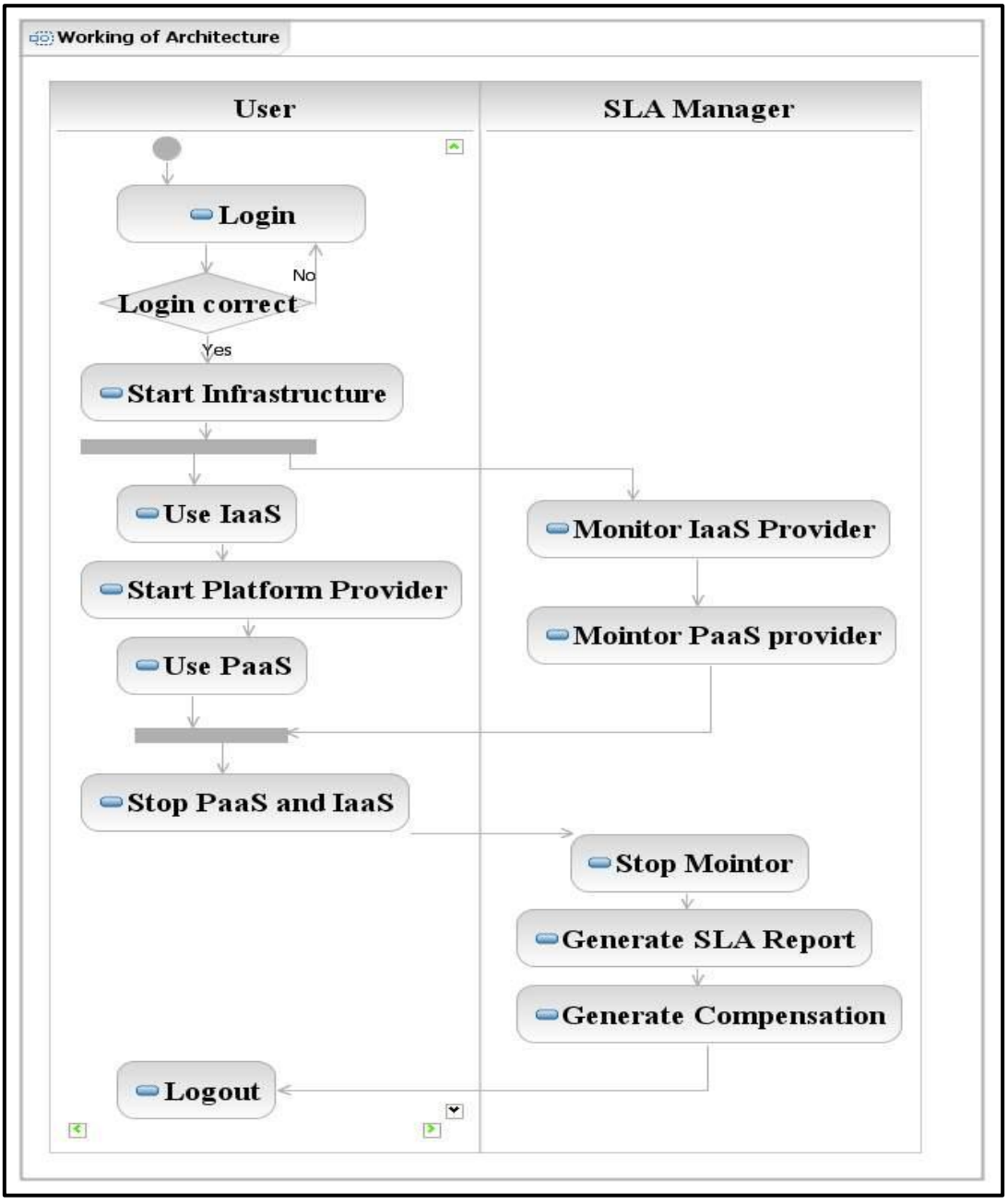

Figure 4.7: Working of Proposed Architecture

Above diagram shows basic activities to start the working of proposed architecture after selecting infrastructure and platform provider.

4.2.3 Class Diagram

Class diagram is most common diagram used in UML. A class diagram shows a set of classes' interfaces, and collaboration and their relationships. Class diagrams are used

to design static view of software system. Class diagrams are foundation for a couple of related diagrams: Component Diagram and Deployment Diagram.

This section shows class diagram of proposed architecture as follows.

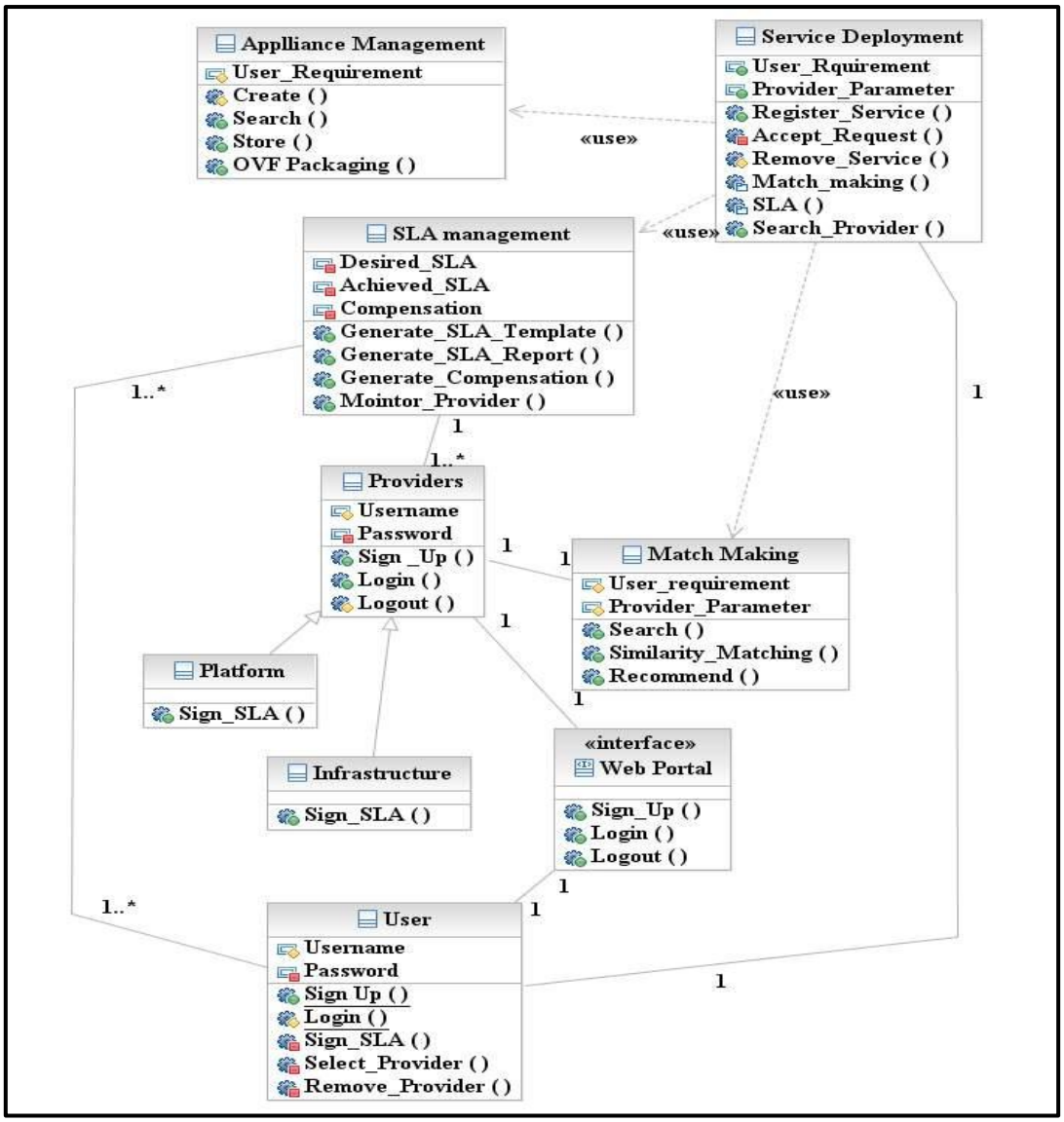

Figure 4.8: Proposed Architecture Class Diagram

4.2.4 Sequence Diagram

A sequence diagram is an interaction diagram that empathize the structural organization of the objects that send and receive messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages, ordering in increasing time, along Y-axis.

This sections shows main sequence diagrams of proposed solution.

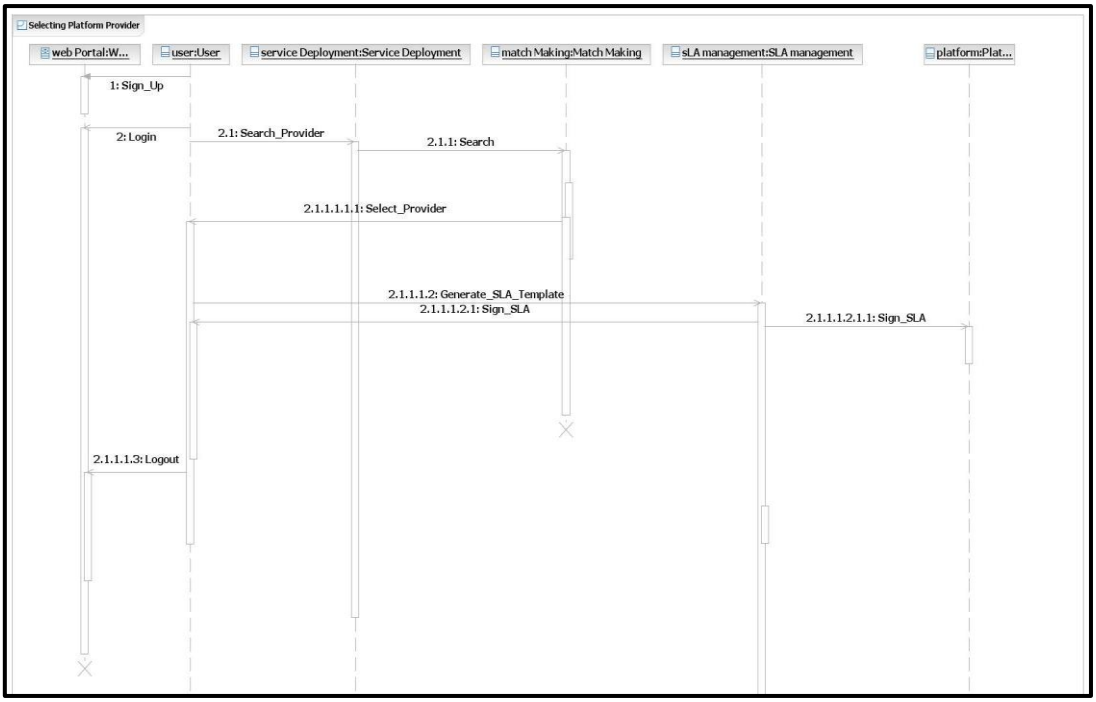

Figure 4.9: Selecting Platform Provider

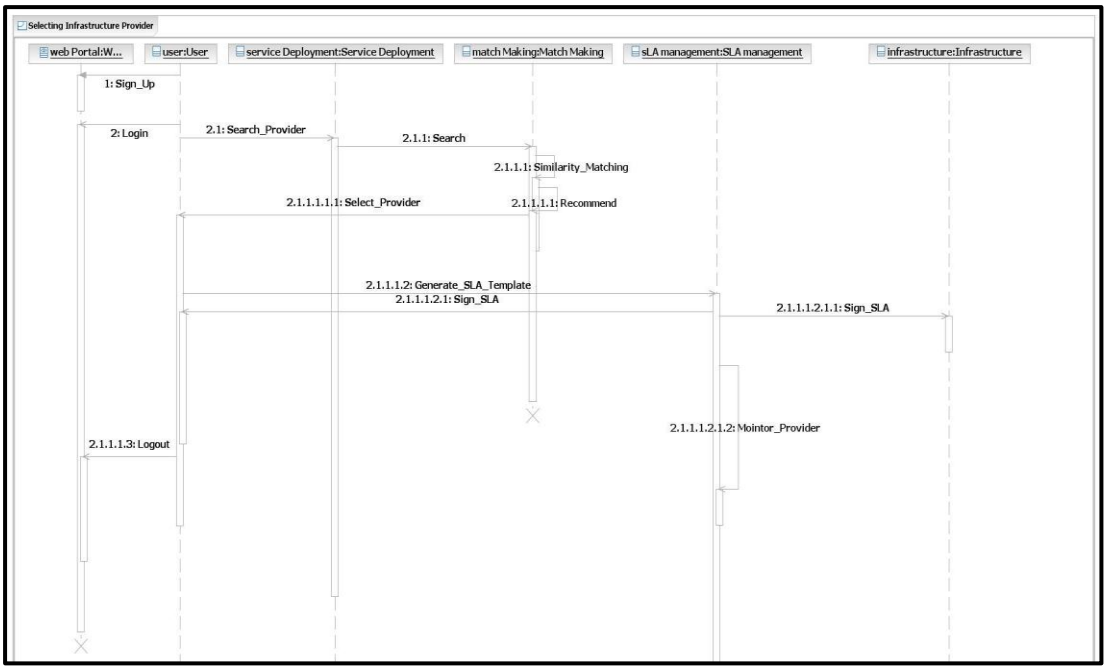

Figure 4.10: Selecting Infrastructure Provider

4.2.5 Automated PaaS Management Architecture Assumptions

There are two assumptions for proposed architecture are:

- Platform provider software is capable of separating infrastructure and user data.
- Providers agree for third party SLA monitoring.

4.3 Automated PaaS Management Architecture

Based on Requirement analysis and UML designing in previous section an Automated PaaS Management Architecture is proposed, as shown in Figure 4.11. This architecture utilizes virtual appliance and OVF packaging to deploy infrastructure of Platform as a Service (PaaS) on third party infrastructure provider. In this section whole architecture is explained in detail component by component.

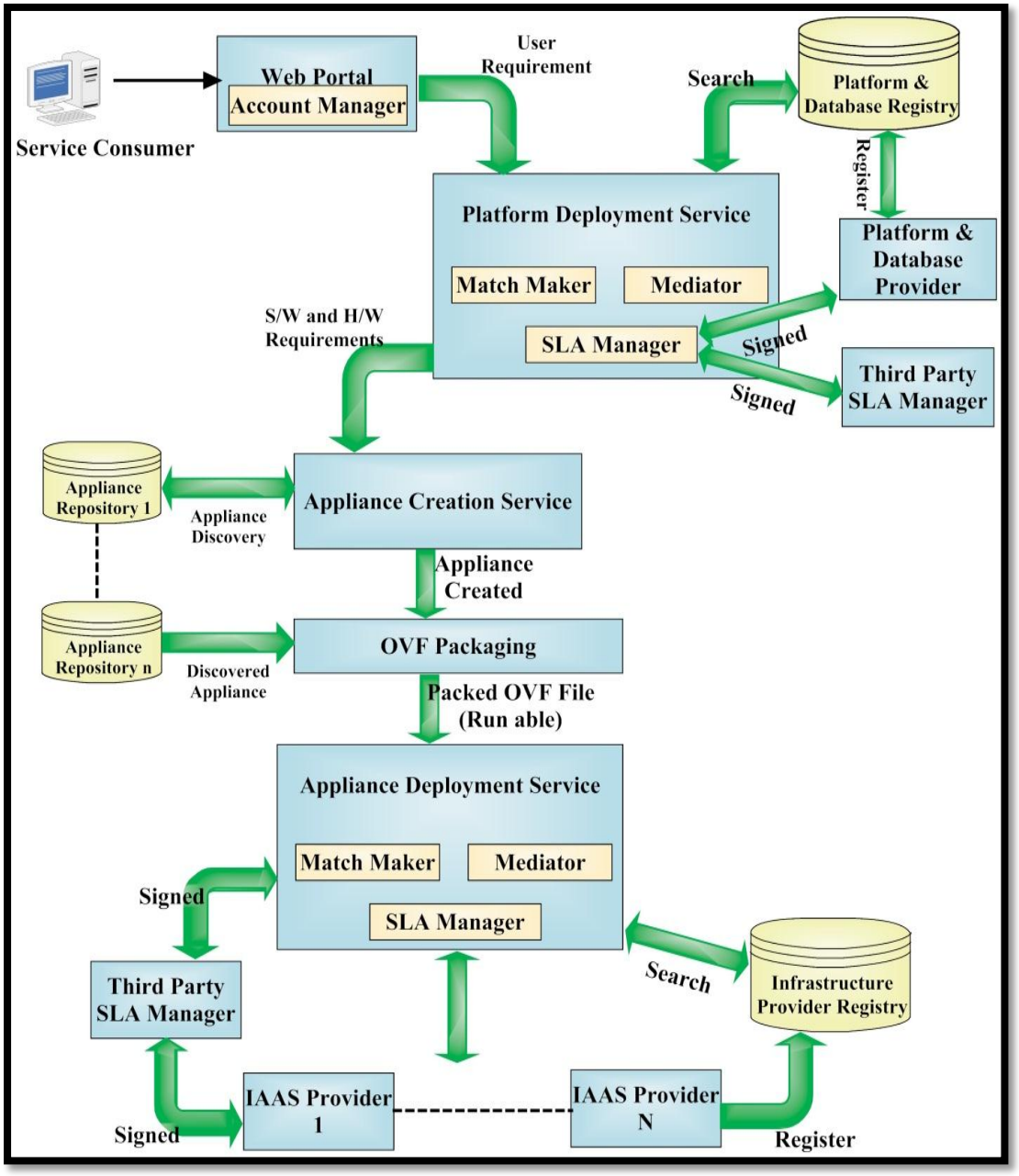

Figure 4.11: Automated PaaS Management Architecture

4.3.1 Web Portal

All services provided by the system are presented via the Web portal to service requesters. In addition, this component provides graphical interfaces to capture user's requirements such as software, hardware, and QoS requirements and contains account manager which is responsible for user management. All requirements are converted into XML format so that it can easily be understood by machine also. These XML format requirements are entered into sematic software for match making. It provides authorization and authentication for users and keeps the history of all users activities in the system.

4.3.2 Platform Deployment Service (PDS)

User requirements registered at web portal are now given to platform deployment service which matches the platform providers suitable to requirements. Platform providers register themselves in platform registry where match maker searches them. PDS acts on interest of user to satisfy his/her QoS by selecting the most desirable platform provider.

Match Making:

In this part, a matching process is proposed that compares the request of the user with the advertisement of the providers. This part ensures user gets most capable provider according to his requirements. One problem with earlier exact keyword matching process is they exclude more capable providers. For example, with the minimum demand, a user requests for a provider who has Window OS. However, in reality, the provider's advertisement is XP professional and this is also a promising provider but exact matching will leave this. In this case suitable approach is to use semantics rather than keywords. Hence, Cloud ontology is designed to describe the resource and use similarity graph algorithm to find capable provider. This process is explained in detail in Section 4.4.

Service Level Agreement (SLA) Manager:

The *pay-per-use* model of Cloud computing is acceptable if users can have minimal guarantees concerning Accuracy, Availability, Capacity, latency and Provision Time. Such requirements can be expressed in Service Level Agreements (SLA) representing some form of contract between users, organizations and providers. These SLAs must

be automatically enforced to the application events and resource usage should be monitored. If there is any SLA violation detected possible counter-measures should be applied. Moreover, to diagnose violations or application misbehaviour, auditing of all details of complex applications must be possible.

In the Automated PaaS Management Architecture SLA is signed by all providers and users and it is monitored by a third party SLA Manager. Proper compensation is also provided to users whose SLA is violated. All of these events are managed by SLA Manager. A compensation algorithm is proposed which is explained in Section 4.4.

4.3.3 Appliance Creation Service (ACS)

After selecting all requirements and signing SLA next step is to create an appliance of desired requirements to deploy it to any infrastructure provider. Appliances are pre-built, pre-configured, ready to run, hypervisor neutral virtual machines.

New appliance can be created according to user requirements using tools like Vmware Studio [57]. Once created appliance with user requirements are stored in database for any future use. These databases are known as Appliance Market Place. Vmware has its own market place for its appliances [58].

Proposed architecture provides two options for appliance creation:

1. Create a new appliance using tools.

2. Search from marketplace for an appliance: For searching an appliance from a market place an Appliance Ontology is created and similarity graph algorithm is used for matching capable appliance as per user requirement. Similarity Graph [46] matching is explained in Section 4.30 above.

4.3.4 OVF Packaging

After acquiring required disk image or their external URL address, user needs to pack them along with other user requirements such as operating system and hardware requirements into a standard format which can run at any hypervisor. The OVF is chosen for this purpose which is adopted by the industry (Vmware, Citrix and rPath) [59]. OVF packaging is done by using ovftool [60].

4.3.5 Appliance Deployment Service (ADS)

Now as Virtual Machine image is created and packaged in OVF. It is ready to deploy on any infrastructure provider. Infrastructure providers register themselves in infrastructure registry where match maker search them. Mediator acts on interest of user to satisfy his/her quality of service (QoS) by selecting the most desirable infrastructure provider.

Match Making:

User requirements as described at web portal are matched with the advertisement of infrastructure provider. An Infrastructure Ontology is created and Similarity Graph matching process is used to match the capable infrastructure provider as per user requirements. Whole matching process is same as describe in Section 4.4. Only difference is that instead of platform provider now it is for infrastructure.

SLA Manager:

After selecting an infrastructure provider all parameters are discussed and signed in an SLA by both User and Provider. An algorithm is proposed for generating compensation if any parameter is violated which is explained in Section 4.5.

4.3.6 IaaS Provider

They are in both fabric and unified resource level and contain resources that can be virtualized. IaaS provider advertises their virtual units on web registry.

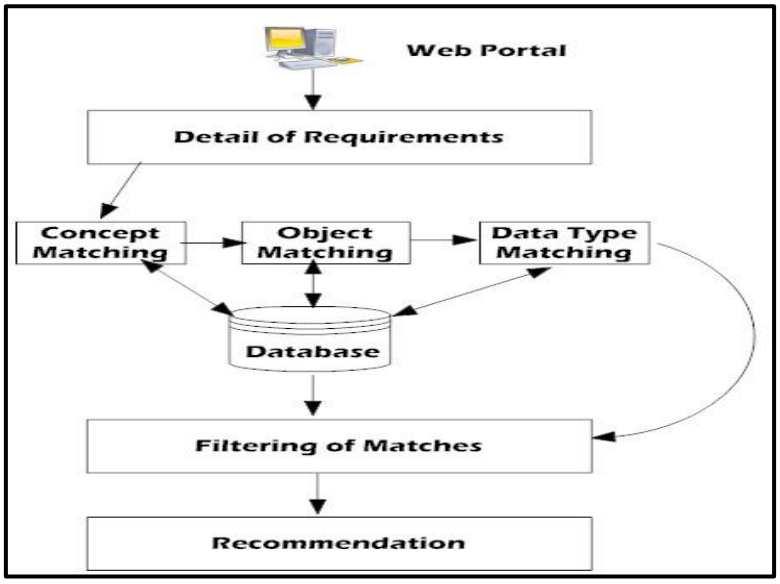

Figure 4.12: Suitable Provider Matching Process

4.4 Match Making Process

After creating an ontology for resource discovery. A matchmaking process is needed so that system can compare two nodes of ontology for their similarity with requirements as explained in Section 4.3.2. Utilizing created ontology, the similarity between two concepts can be determined using process shown in Figure 4.12.

This system works as follows. User sends its requirements from a web portal. According to requirements three matchings are done viz. Concept Matching, Object Matching and Data type Matching [50]. Using a semantic score all providers whose score is less than threshold score are filtered out. Semantic score is calculated using Equation III.

$$Sim(x,y) = Sim_{con}(x,y) + Sim_{obj}(x,y) + Sim_{data}(x,y) \quad \text{..... (III)}$$

- Concept Matching:
  First type of matching done for finding similarity between x and y is concept matching. Concept Matching is done by using Similarity Graph [46] as follows:

$$Sim_{con}(x,y) = \frac{|Super(X) \cap Super(Y)|}{|Super(X)|} \quad \text{... ... ... (III)}$$

  Where *X* and *Y* are the most specific concepts that individuals *x* and y belong to, respectively, and *Super*(*P*) (respectively, *Super*(*Q*)) is a set of all reachable super-concepts from concept *P* (respectively, concept *Q*).

- Object Matching:
  Object Matching can be determined as follows:

$$Sim_{obj}(p,q) = \frac{\sum_{(x,y)\in U} Sim(x,y)}{|O(p)|} \quad \text{... ... ... (IV)}$$

$$where, \qquad U = \{(x,y)|(p,r,x) \in O(p), (q,r,y) \in O(q)\}$$

  where *O*(*p*) is a set of triples that contain the object properties of the individual *p*, and *q* is the subject. Each triple consists of (1) the subject, (2) a predicate, and (3) an object value to express the ontology. For instance, if user want to express the individual '*Provider1*', which has the property '*hasDBMS*', and its value

'*Oracle*', user can simply express using a triple as '*(Provider1,hasDBMS,Oracle)*'. *U* is the set of object values that has the common predicate *r* of individuals *p* and *q* in each triple *O*(*p*) and *O*(*q*) , respectively.

- Data type Matching:

  Data type matching can be determined as follows:

$$Sim_{data}(p,q) = \frac{\sum_{(x,y,r)\in V} Comp(x,y,r)}{|D(P)|} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (VI)$$

$$where, \quad V = \{(x,y,r)|(p,r,x) \in D(P), (q,r,y) \in D(Q)\}$$

$$Comp(x,y,r) = 1 - \frac{|x-y|}{MAX_{distance}(x,r)}$$

$$MAX_{distance}(x,r) = max_{i\in I(r)}(|x-i|)$$

$$I(r) = \{i|(s,r,i) \in Ontology\}$$

  where *D*(*p*) is a set of triples that contains the datatype properties of the individual *p* and *p* is the subject. Each triple consists of (1) the subject, (2) a predicate, and (3) a datatype value to express the ontology. For instance, if user wants to express the individual '*Provider2*', which has the property '*hasNetworkBandwidth*', and its value '*1000'*, user can simply express using a triple as '(*Provider2*, *hasNetworkBandwidth*, *1000*)'. *V* is a set of datatype values that has the common predicate *r* of individuals *p* and *q* in each triple *O*(*p*) and *O*(*q*) respectively. With each of the elements in *V*, *Comp*(*x*, *y*, *r*) , which is the similarity between datatype values *x* and *y* over predicate *r* is determined.

4.5 Service Level Agreement (SLA) Violation Algorithm

In Automated PaaS management Architecture SLA manager finds any violation from signed SLA and calculates its compensation, explained in Section 4.3.2. Here, an algorithm is proposed to calculate compensation. This section explains proposed algorithm in detail.

In context to architecture, following metrics are considered [51]:

1. **Accuracy:** It is concerned with the error rate of the service. It is possible to specify the average number of errors over a given time period. Accuracy is measured in 1-10 range with 10 being highest error rate.

2. **Availability**: It is concerned with the mean time to failure for services. It is typically measured by the probability that the system will be operational when needed. It is from 1 to 10. 10 being highest available

3. **Capacity**: It is the number of concurrent requests that can be handled by the service in a given time period. It is possible to specify the maximum number of concurrent requests that can be handled by a service in a set block of time. it is just numeric value of concurrent request.

4. **Latency**: It is concerned with the maximum amount of time between the arrival of a request and the completion of that request. it is measured in minutes.

5. **Provision Time**: It is concerned with amount of time taken for making a new client operational. It is measured in minutes.

These metrics are entered into a matrix as shown below:

$$\begin{bmatrix} A & D & PV & C & T \\ D & A & PV & C & T \\ D & A & PV & C & T \\ A & D & PV & C & T \\ A & D & PV & C & T \end{bmatrix} \begin{matrix} Accuracy \\ Availability \\ Capacity \\ Latency \\ Provision\ Time \end{matrix}$$

where,

$$A \rightarrow Achieved, D \rightarrow Desired, PV \rightarrow Persistence\ Value, C \rightarrow Cost, T \rightarrow Time$$

- Achieved: It is value of metrics that is actually achieved or provided by provider. It is entered by Third party SLA monitoring organization.

- Desired: It is value that provider promise to provide to Cloud user. It is recorded when provider and user signed SLA.

- Persistence Value (PV): Every desired value can't be achieved in real time application because of many reasons. PV provides the probability to which desired value will not be achieved. Lower the PV value higher price will be for same service. For example, Desired Capacity value is 100 so 100 concurrent request can be managed by system and if PV value is 10 means there is probability that capacity value would be 90 but not less than 90. Compensation will not be provided if achieved metric is under persistence value.

- Cost: It is amount of cost to be paid in compensation for each unit of PV. For example, Desired Capacity is 100, PV value is 10, Cost/PV= 100$ so if achieved value is 50 than 100-50=50. It is 5 unit of PV so compensation will be 500$. This value is in Dollars.

- Time: It is amount of time after which each metrics will be checked and compensation is generated. By default this value is in months.

As Availability and Capacity are desired higher so their first column contains Desired and Second Achieved. Achieved value is metric that provider actual provide whereas Desired is metric that provider signed to provide. Persistence Value is unit in which Achieved and Desired metric can vary. Cost is amount of compensation provider has to pay to violated per unit of PV. Time is amount of time in which this metric will be checked and counter is reset.

Proposed SLA violation compensation algorithm is explained below using above matrices.

Algorithm 1:

1. Value of each parameters A, D, PV, C, T are entered in a matrices named SLA[5][5] when user and provider signed their SLA.
2. After the completion of stated Time Period value of achieved metric is entered.
3. $for(\ i = 1; i \leq 5; i++)$
{
if(T = 0)
{

```
            if( SLA[i][0] – SLA[i][1] >= 0)
                print No Violation of SLA so no Compensation
        else
            {
                Violated = SLA[i][1] – SLA[i][0]
                if( Violated > SLA[i][2])
                {
                    Violated = violated % SLA[i][2]
                    Compensation = Violated * SLA[i][3]
                    print Compensation
                }
                else
                    Prit No compensation
            }
        }
}
```

4. Reset Time

Using Algorithm 1, compensation for SLA violation can easily be found.

Example.

Suppose initial matrix is as follows:

$$\begin{bmatrix} 0 & 1.5 & .5 & 10 & 5 \\ 9.5 & 0 & .5 & 10 & 5 \\ 100 & 0 & 10 & 10 & 5 \\ 0 & 15 & 5 & 10 & 5 \\ 0 & 15 & 5 & 10 & 5 \end{bmatrix}$$

After 5 months suppose following SLA are achieved:

$$\begin{bmatrix} 1.7 & 1.5 & .5 & 10 & 5 \\ 9.5 & 9.7 & .5 & 10 & 5 \\ 100 & 50 & 10 & 10 & 5 \\ 10 & 15 & 5 & 10 & 5 \\ 21 & 15 & 5 & 10 & 5 \end{bmatrix}$$

So compensation will be:

Accuracy: 1.5-1.7= -.2, PV= .5 so no Compensation required

Availability= 9.7-9.5= .2 so no compensation

Capacity= 50-100= -50, PV= 10, Units for compensation = 50%10=5, Compensation= 10*5= 50 $

Latency= 15-10=5, so no compensation

Provision time= 15-21= 7, PV=5, units for compensation= 7%5=1, Compensation= 10*1=10$

So, total Compensation= 50 + 10= 60$

4.6 Conclusion

In this chapter an architecture namely Automated PaaS Management is proposed by analysing all requirements. This architecture solves some portability issues in PaaS Environment. Semantics is used for provider search according to users (QoS) requirements. This assigns most capable provider to user at first time and reduces the need of portability. An SLA violation compensation algorithm is also proposed. This algorithm is very simple and easy to calculate compensation. Next section implements the proposed architecture.

Chapter 5

Implementation

This chapter focuses on experimental setup and implementation of proposed architecture. First experimental setup is explained covering all hardware requirements for implementation. Next tools used for implementation are explained briefly. Lastly implementation and result of each component in proposed architecture is shown. Results show how virtual machine is successfully migrated to another machine with heterogeneous hypervisors achieving portability.

5.1 Experimental Setup

Experiments are conducted to verify the proposed architecture. Four virtual machines of 3 GB Ram, 2-core 2 Ghz processor and 160 Gb of hard disk with Window XP installed are used as infrastructure provider. A server with configuration of 8 GB RAM, 8-core 2 Ghz processor and 320 Gb hard disk is used as platform provider. Aneka 3.0 is installed on the Server. Each infrastructure provider has 2 virtual machines. One virtual machine is used to keep all user data along with application and second virtual machine is used for using its resources.

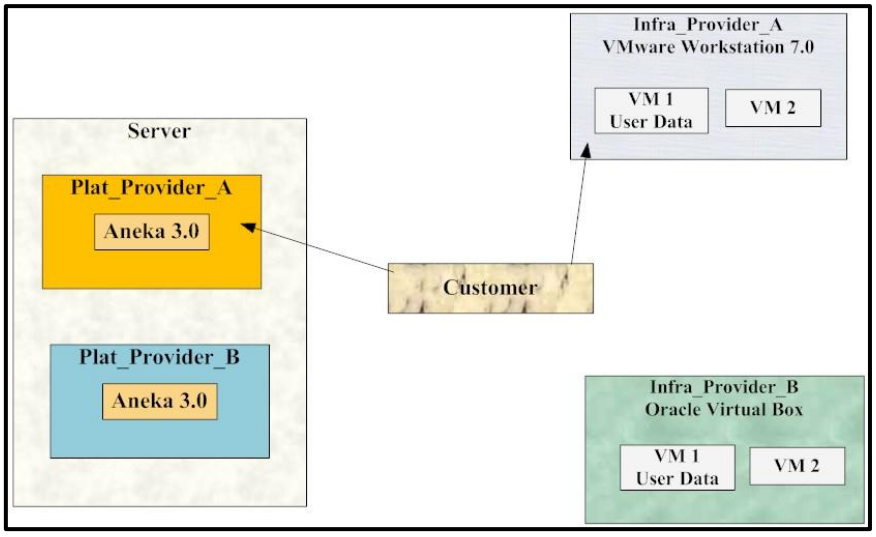

Figure 5.1: Initial Setup of Proposed Architecture

In experimental setup two infrastructure providers are used, named Infra_Provider_A and Infra_Provider_B, Infra_Provider_A is using Vmware Workstation 7.0 for hypervisor and Infra_Provider_B is using and open source Oracle Virtual Box as a

hypervisor. There are two different providers of Aneka 3.0 platform, named Plat_Provider_A and Plat_Provider_B, installed on Server. An application, Convolution, is developed on Aneka 3.0 which User_A is using, as shown in Figure 5.1.

5.2 Tools Used

This section gives some brief of tools used to implement proposed architecture.

5.2.1 Protégé 4.0

Protégé [61] is open source software develop by Stanford University to create ontologies in Ontology Web Language (OWL) and Relational Database Framework (RDF). It is a java based tool. Key features provided by Protégé 4.0 are as follows [43]:

- Creating, loading and saving both Ontology Web language (OWL) and Relational Database framework (RDF) ontologies.
- Provide functionality and interface to create classes, their properties and relationship between two classes.
- Reasoner is provided by Protégé 4.0 which is used to test whether or not one class is subclass of another class. By performing such test on the class in ontology it is possible for reasoner to compute the inferred ontology class hierarchy. Another service offered by reasoner is consistency check.
- Execution of different queries on ontology. These executed queries can also be saved for future use.

There are several protégé versions available like Protégé 4.2 beta, Protégé 4.1.3, Protégé 3.5 beta, Protégé 3.4.8, Web Protégé etc. Latest version is Protégé 4.2.

5.2.2 VMware Studio 2.0

VMware Studio [29] is an integrated development tool that takes existing software applications and packages them into virtual machines and vApps that are ready to run and optimized for VMware product platforms. VMware Studio can build both Linux-based VMs and Windows-based virtual machines and vApps, running single tier or multitier applications.

Key features provided by VMware Studio 2.0 are:

- Provider functionality and interface to build virtual appliance and virtual Apps.
- Contains build profiles for VMware vFabric Application Director appliances based on CentOS, RHEL, SLES, and Ubuntu.
- It automatically resolves package dependencies.
- VMware Studio helps you package multitier applications into a vApp that runs efficiently under VMware vSphere.
- VMware Studio can use different VMware platform products to generate builds of virtual appliances. Supported platform products include vCenter Server, ESX/ESXi hosts, and VMware Workstation.
- It provides plugin for Eclipse which helps with the iterative process of coding and testing, especially for developers comfortable with Eclipse.
- VMware Studio allows developers to avoid building a virtual machine from scratch each time. After user build a virtual machine with VMware Studio, or obtain a qualifying Linux virtual machine from elsewhere, it can serve as input for a subsequent build (CLI only).

Figure 5.2 shows architecture of VMware studio as given by VMware.

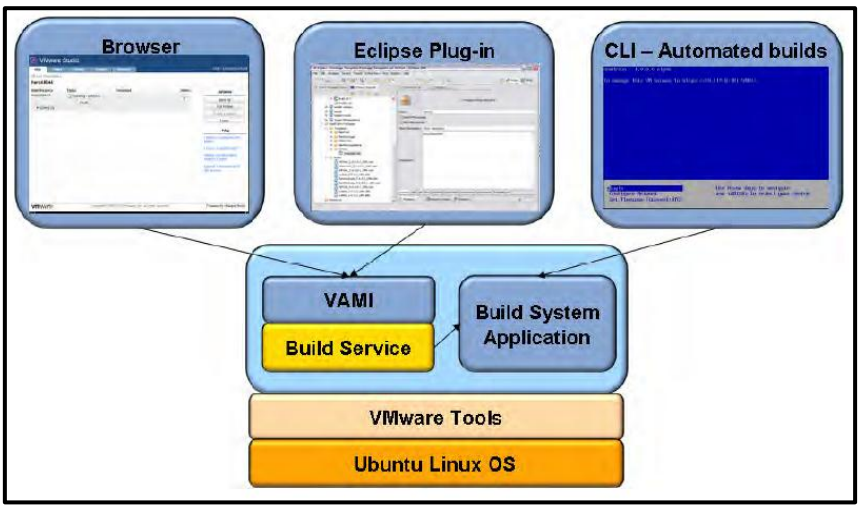

Figure 5.2: VMware Studio 2.0 Architecture [29]

5.2.3 OVF Tool

Open Virtualization Format (OVF) is an industry standard that describes metadata about virtual machine images in XML format. OVF Tool is a command-line utility that enables a user to import and export OVF packages to and from a wide variety of

hypervisors. User can use OVF Tool to distribute and import virtual machines and vApps. For example, you can create a virtual machine within VMware vSphere™, and use OVF Tool to export it into an OVF package for installation, either within your organization or for distribution to other organizations. OVF facilitates the use of vApps, which consist of preconfigured virtual machines that package applications with the operating system that they require.

Using VMware Studio 2.0 user can build new appliances but using OVF Tool user can convert any virtual machine into an appliance with all applications installed.

OVF Tool 1.0 provides the following key features [62]:

- Includes full OVF 1.0 support and backward-compatible mode for importing existing OVF 0.9 packages.
- Supports both import and generation of OVA packages (OVA is the portable virtual machine format from XenSource).
- Accesses OVF sources using HTTP, HTTPS, or FTP, or from a local file.
- Show information about the content of any source in probe mode.
- Provides an optional output format to support scripting when another program calls OVF Tool.
- Signs OVF packages and validates OVF package signatures.

5.2.4 Aneka 3.0

Aneka [63] [64] is a Cloud Application Development Platform (CAP) for developing and running compute and data intensive applications. As a platform it provides users with both a runtime environment for executing applications developed using any of the three supported programming models, and a set of APIs and tools that allow you to build new applications or run existing legacy code.

An Aneka Cloud is composed of a collection of services deployed on top of an infrastructure. This infrastructure can include both physical and virtual machines located in your local area network or Data Centre. Aneka services are hosted on *Aneka Containers* which are managed by *Aneka Daemons*. An Aneka Daemon is a background service that runs on a machine and helps you to install, start, stop, update and reconfigure Containers.

A key component of the Aneka platform is the Aneka Management Studio, a portal for managing your infrastructure and clouds. Administrators use the Aneka Management Studio to define their infrastructure, deploy Aneka Daemons, and install and configure Aneka Containers. Figure 5.3 below shows a high-level representation of an Aneka Cloud, composed of a Master Container that is responsible for scheduling jobs to Workers, and a group of Worker Containers that execute the jobs. Each machine is typically configured with a single instance of the Aneka Daemon and a single instance of the Aneka Container.

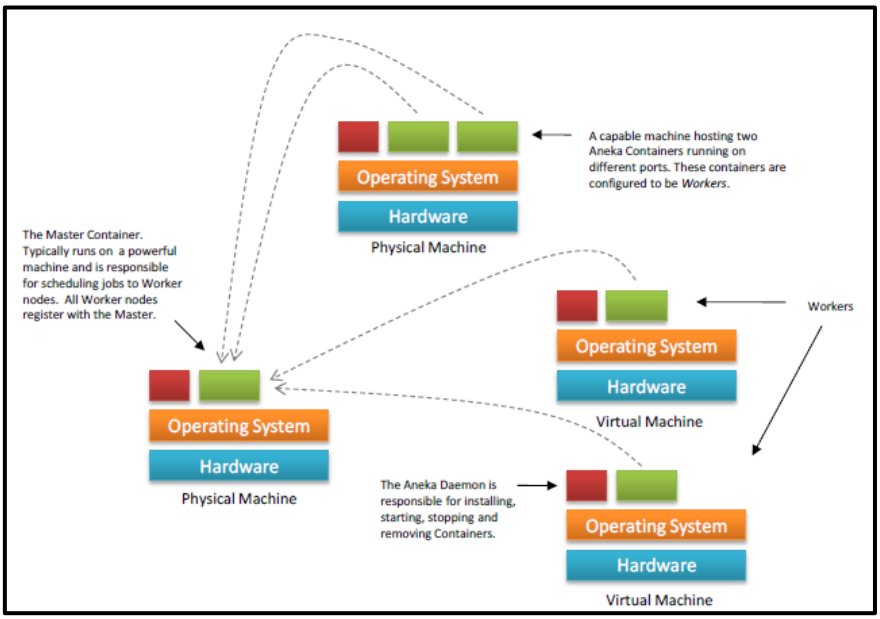

Figure 5.3: High Level View of Aneka Cloud [63]

5.3 Implementation

This section shows implementation of proposed architecture using tools explain in above section.

5.3.1 Website:

As shown in Figure 5.4 below, website for Sign Up and login of different providers and users have been created. Profile of all providers and users is maintained and managed by using this website.

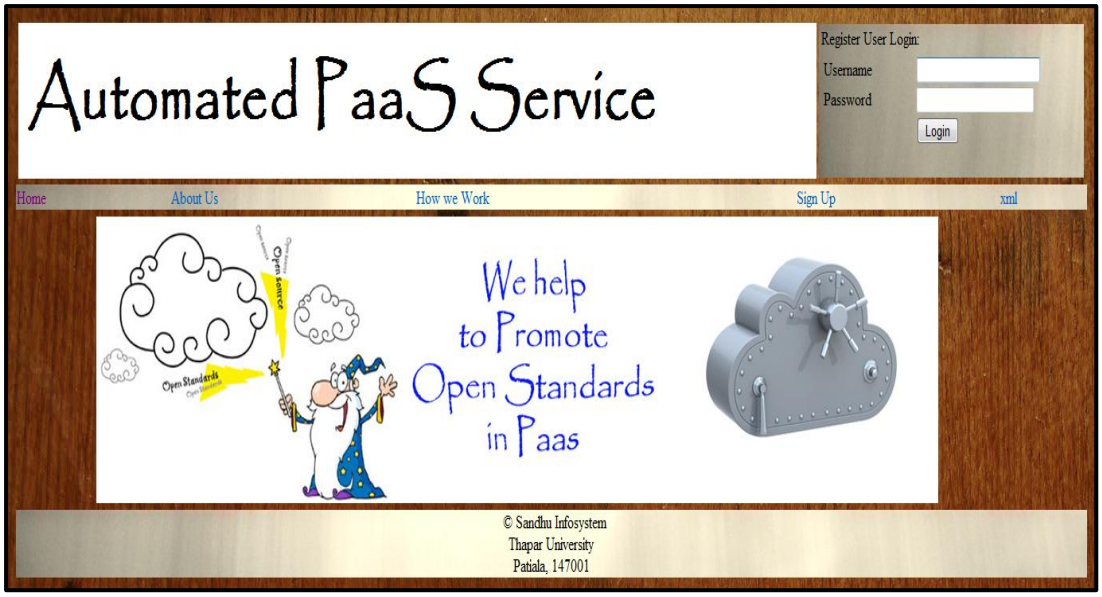

Figure 5.4: Website Home Page

SignUp page as shown in Figure 5.5, provides an option at the end to choose type of user for this website i.e. Platform Provider, Infrastructure Provider, Service Consumer or SLA Manager. One user can act on one role only for example Platform Provider can't act as Infrastructure Provider in the same profile.

Figure 5.5: Sign Up Page for Users

User search option page as shown in Figure 5.6, allow user to search through the platform provides registered. These all options are fed into ontology and result is shown back to user.

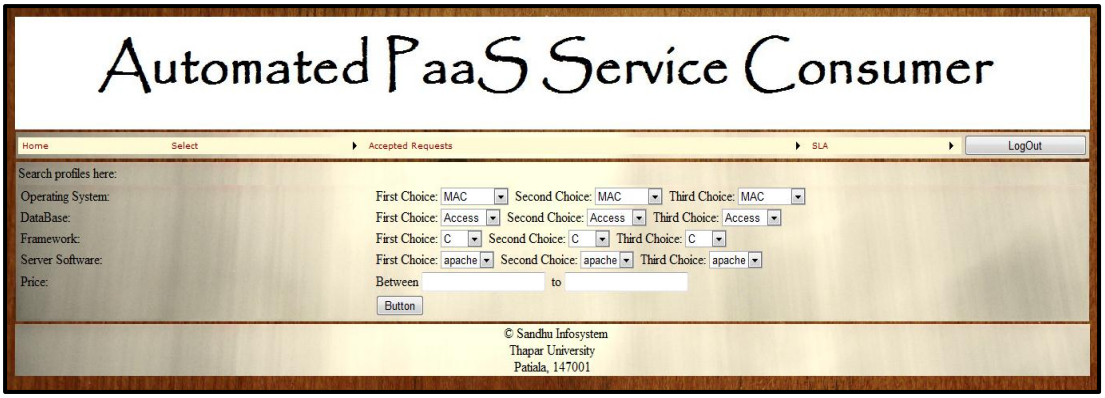

Figure 5.6: User Search Option for Providers

As shown in Figure 5.7, single Infrastructure provider can create multiple profiles. This webpage allow provider to enter profile information. This profile information is converted into an XML file and fed into ontology using XMLtab in protégé 4.0. Same working is done in case of Platform Provider profile registering, as shown in Figure 5.8.

Figure 5.7: Infrastructure Provider Enter Profile

Figure 5.8: Platform Provider Registering Profile

5.3.2 Discovery Process:

An ontology for semantic searching of capable provider to user requirements has been created. Three ontologies are created each for Platform Provider, Infrastructure Provider and Appliance Discovery.

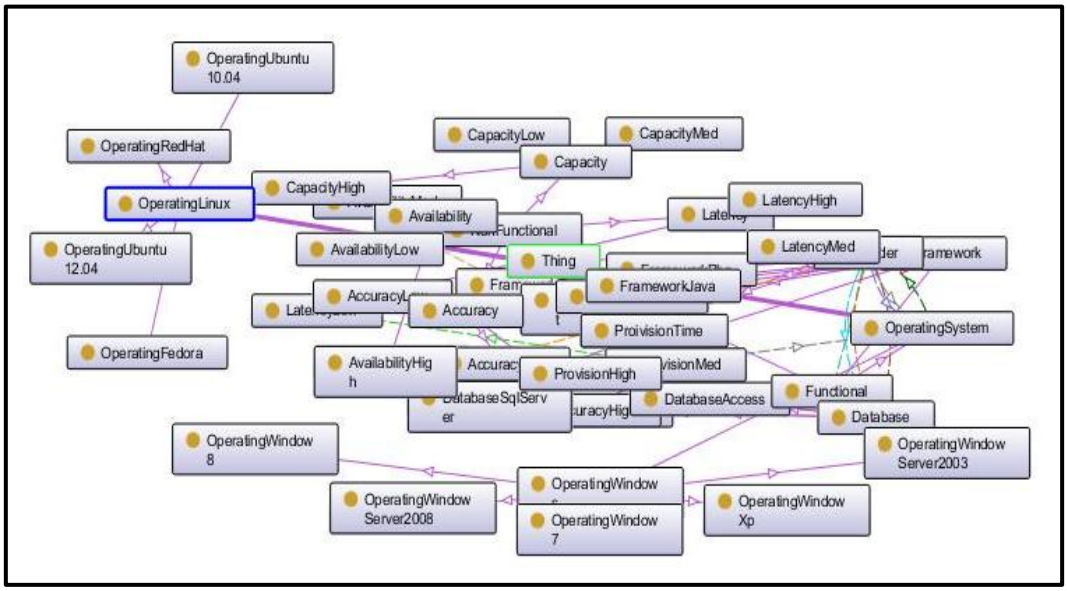

Figure 5.9: Platform Discovery OntoGraph

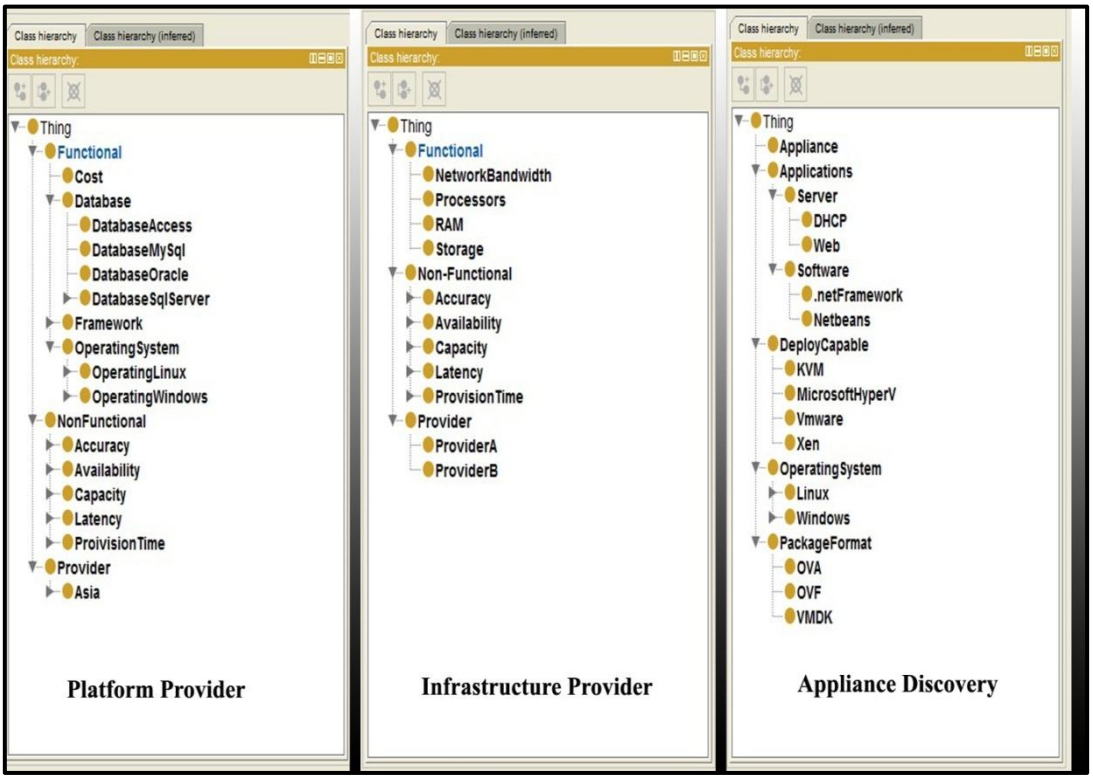

Figure 5.10: Provider and Appliance Discovery Ontologies

Figure 5.9 shows onto graph of platform ontology. This shows semantic web of ontology created. As shown in Figure 5.10 left hand side ontology, search is made on basis of Functional and Non-Functional requirements mentioned by user in the website mentioned above. In order to get result DL query is used. Result of DL query is name of provider, as shown in Figure 5.11. In the same manner two more ontologies are created for Infrastructure Provider Discovery and Appliance Discovery. Their OntoGraph are shown in Figure 5.12 and Figure 5.13 respectively. Infrastructure Provider and Appliance discovery ontologies are shown in Figure 5.10. DL queries for Infrastructure and Appliance are implemented in the same manner as shown in Figure 5.11.

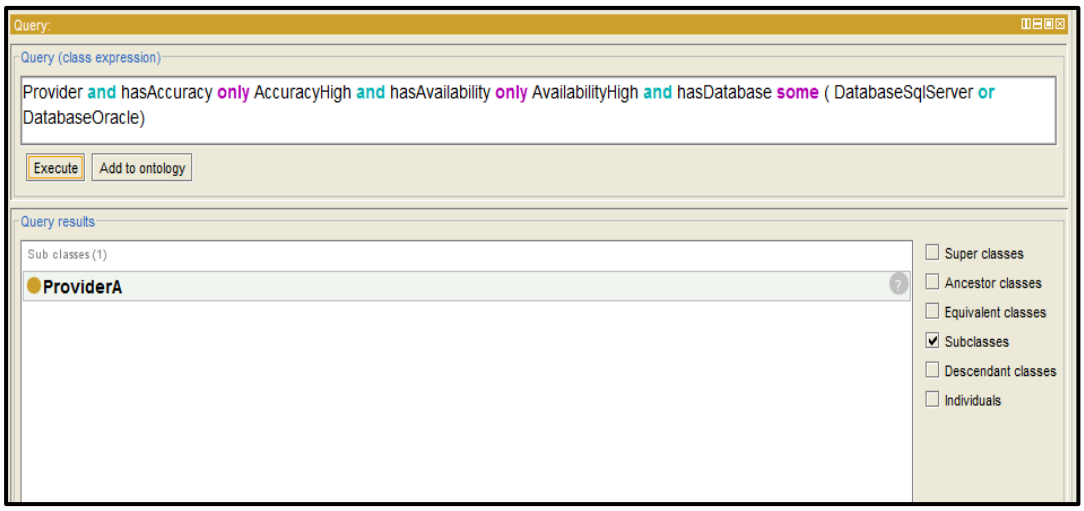

Figure 5.11: DL Query on Platform Provider Ontology

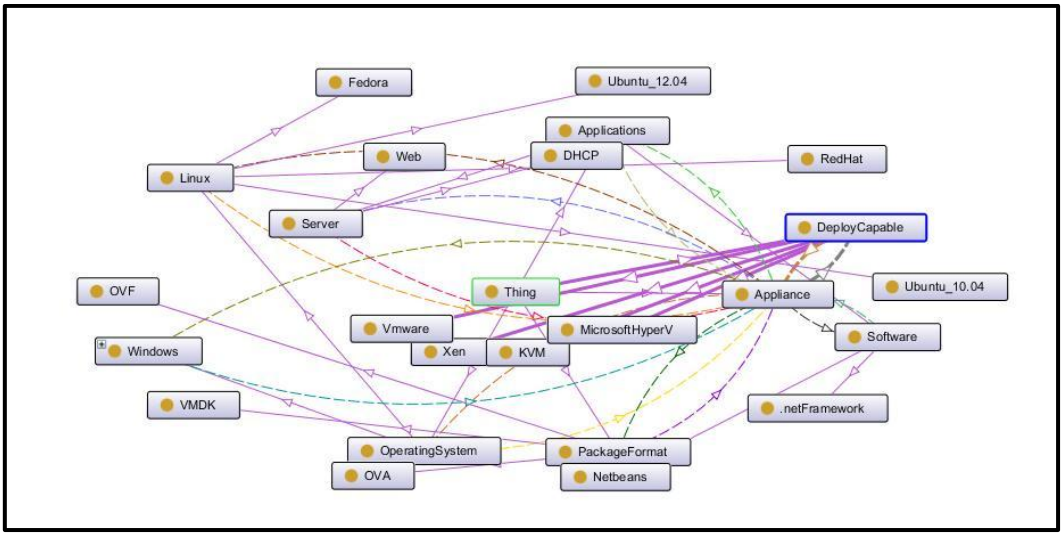

Figure 5.12: Appliance Discovery OntoGraph

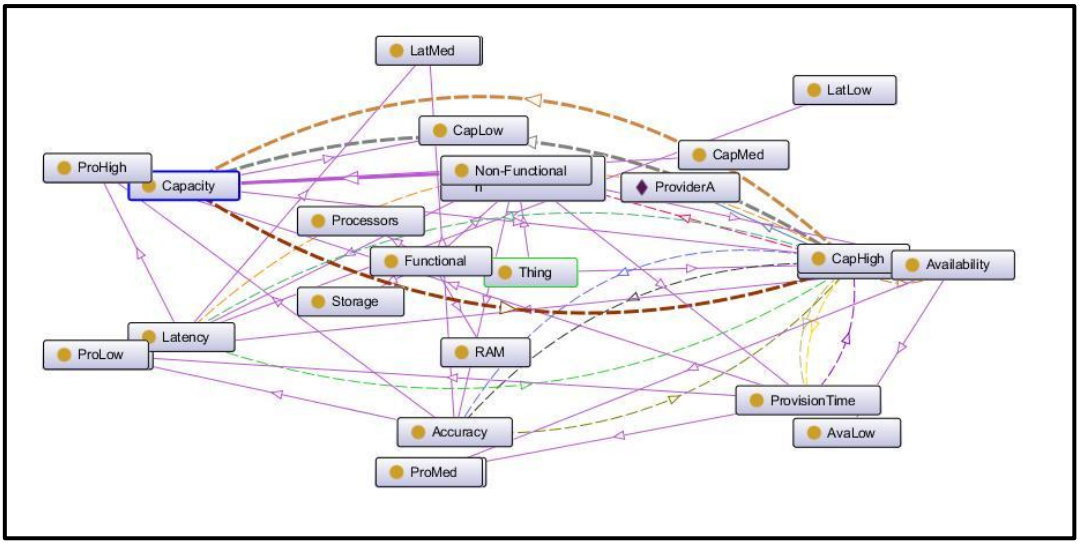

Figure 5.13: Infrastructure Provider OntoGraph

5.3.3 Virtual Appliance Creation:

After searching is complete as stated in above section, a capable platform and infrastructure provider is selected. Now user virtual machine which will be deployed on infrastructure provider should be converted into a virtual appliance. This section shows hoa a virtual machine can be converted into a virtual appliance. .net Framework is pre-installed in these appliances. Following are steps to create a fresh virtual appliance.

1. Download VMware Studio from VMware studio website.
2. Install VMware studio on VMware Workstation 7.0.
3. As shown in Figure 5.14, screen will appear showing your ip where to locate VMware studio GUI. In this case it is http://192.168.178.129/.
4. Open any web browser and enter the provided ip, as shown in Figure 5.15. A window will appear to login into VMware Studio. Username is root and password will be set by user while installing VMware Studio.
5. After Login create a new profile and select the template for which virtual appliance has to make, shown in Figure 5.16 and Figure 5.17.
6. Packages such as .net framework can be added into the appliance using Eclipse VMware studio Explorer, shown in Figure 5.18.
7. Create a new package repository to upload all the required packages, as shown in Figure 5.19. Upload all required package into new created repository on VMware Studio 2.0.

8. Fresh Virtual appliance will be created with two files as shown in Figure 5.27.

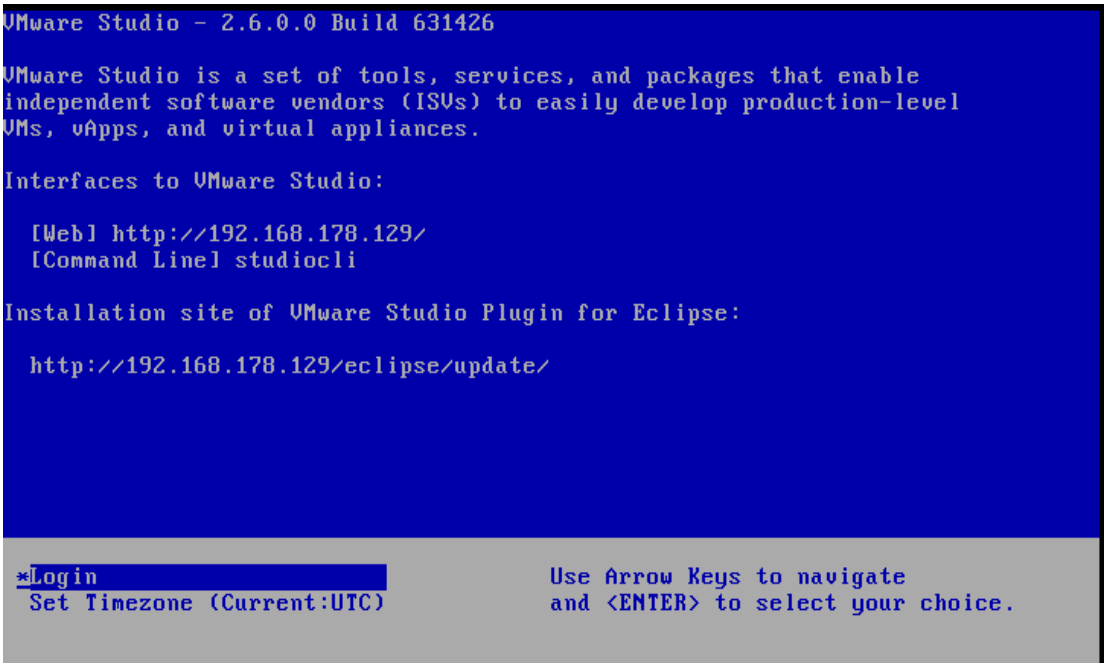

Figure 5.14: VMware Studio Start Up Screen

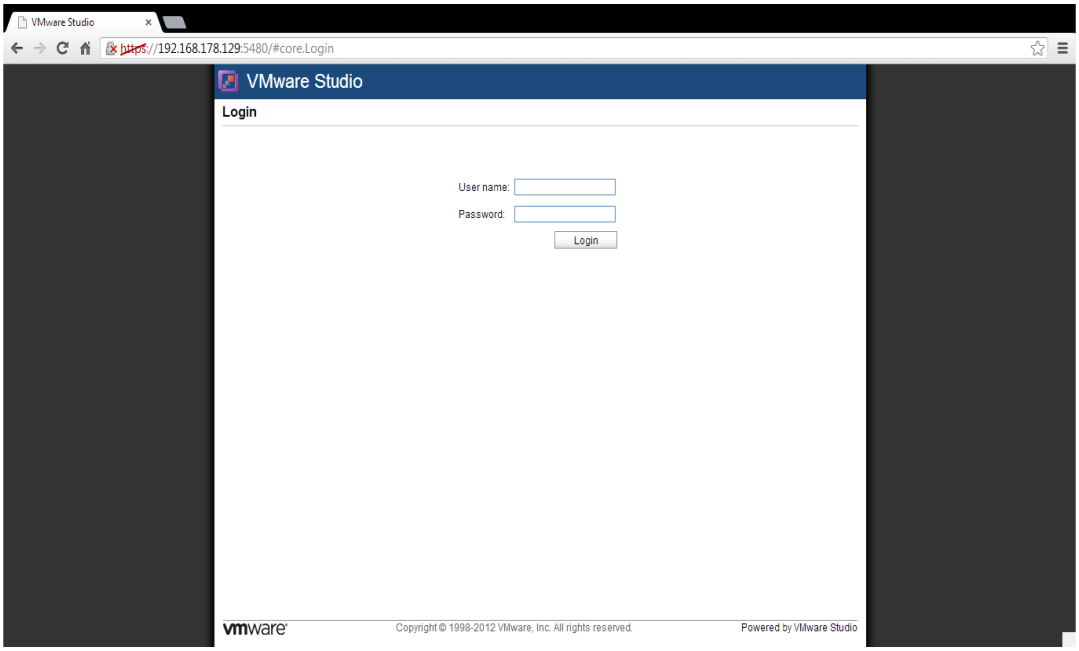

Figure 5.15: VMware Studio Web Browser Welcome Screen

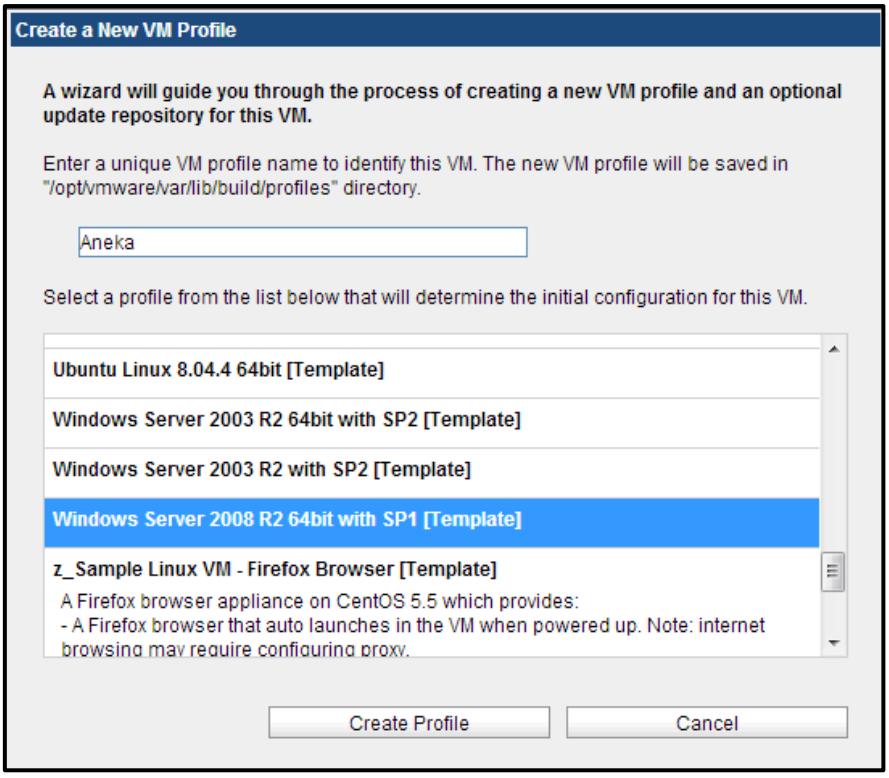

Figure 5.16: Selecting Template for creating Virtual appliance

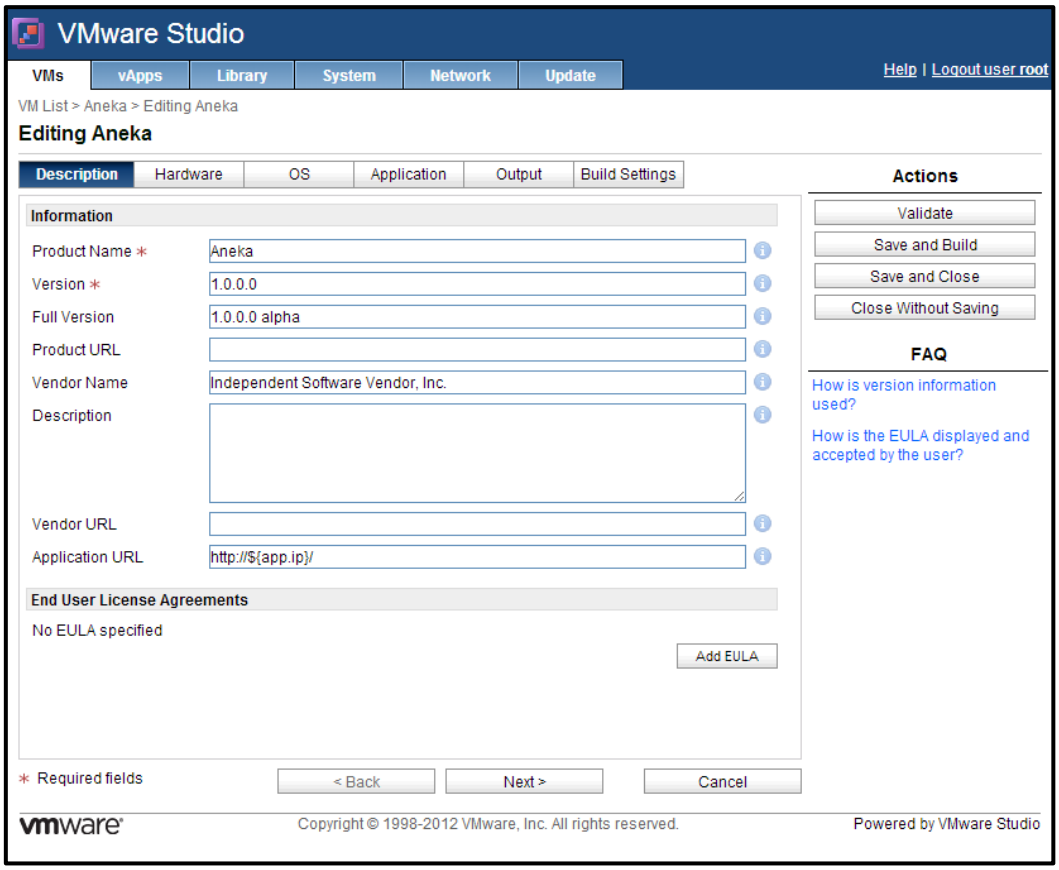

Figure 5.17: VMware Studio Create Profile

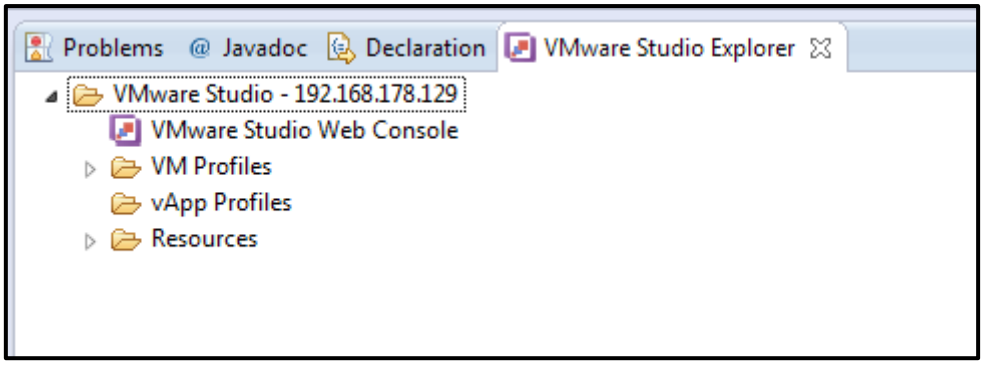

Figure 5.18: Eclipse VMware Studio Explorer

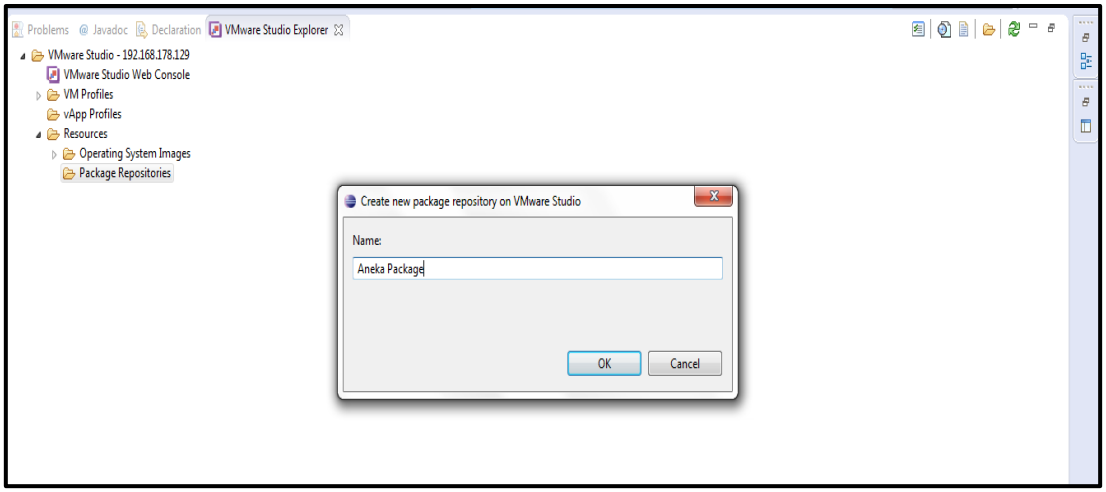

Figure 5.19: Add Package Repository

5.3.4 Service Level Agreement Violation Compensation

As discussed in above sections, matrix based SLA violation compensation algorithm has been proposed. This algorithm includes SLA Manager to enter all the desired SLA as signed between both User and Provider. Then after the completion of desired time this algorithm calculated compensation automatically and give message to User, Provider and SLA Manager. This algorithm is implemented in Asp.net 3.5 framework in the main website where User and Provider Register. Below snapshots shows implementation.

As shown in Figure 5.20, SLA matrix is entered by SLA Manager after User and Provider agree to a SLA.

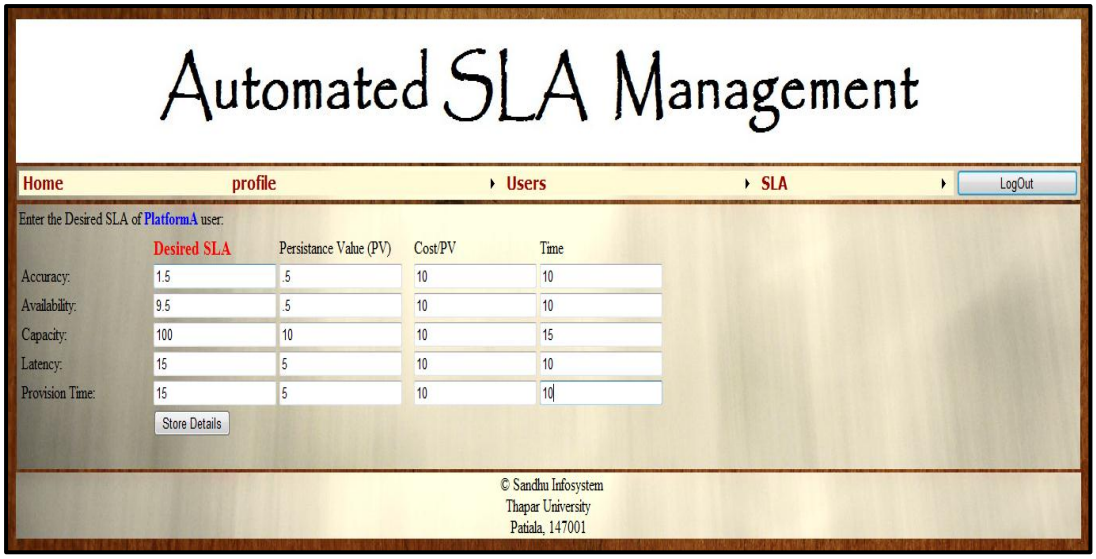

Figure 5.20: Enter the SLA Matrix

After the time is over SLA manager enters the achieved SLA of a particular agreement, depicted in Figure 5.21.

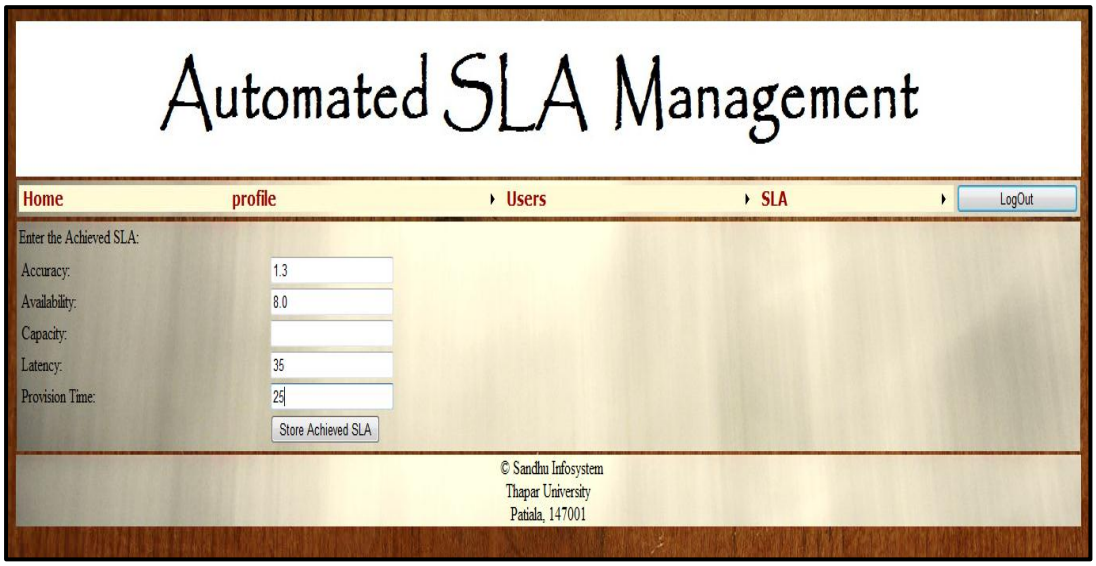

Figure 5.21: Achieved SLA

After submitting the SLA website automatically shows the compensation in respective metrics, shown in Figure 5.22.

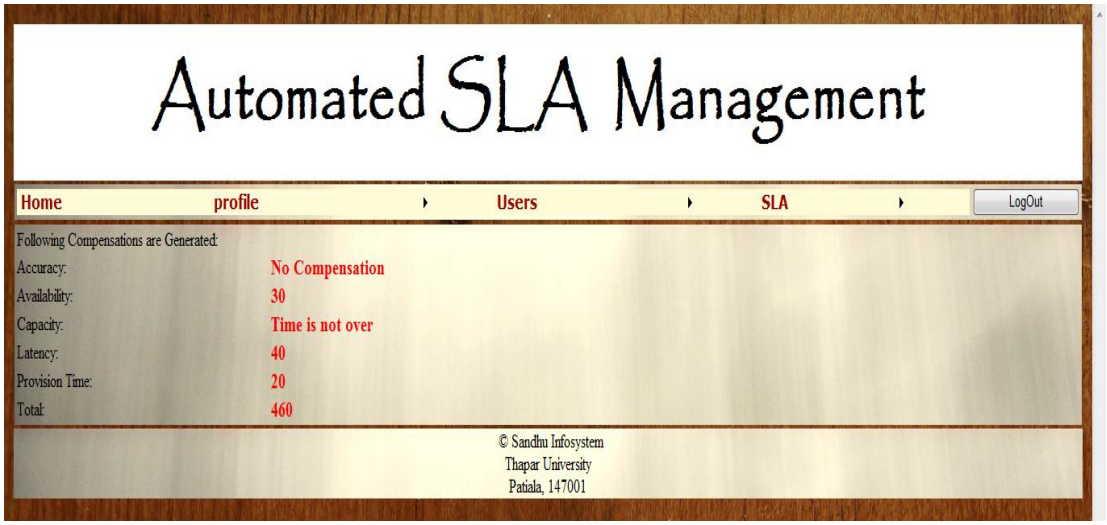

Figure 5.22: Compensation of SLA Violation

5.3.5 Initial Setup:

As explain in Section 5.1, User is registered to Platform_Provider_A for platform and Infra_Provider_A for infrastructure. As shown in Figure 5.23, two Infrastructure groups named VMware Workstation and Oracle Virtual Box have been created. VMware Workstation is Plat_Provider_A.

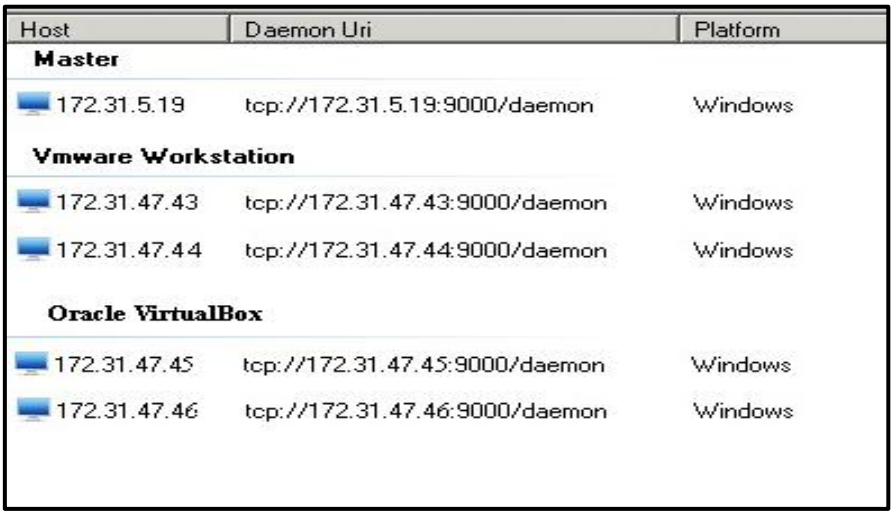

Figure 5.23: Aneka Setup

Credentials are set for Plat_Provider_A in Aneka as shown in Figure 5.24. Using these credentials in Convolution application, as shown in Figure 5.25, user connects and uses Aneka services. This is Initial Setup. Below different cases according to our proposed architecture are discussed.

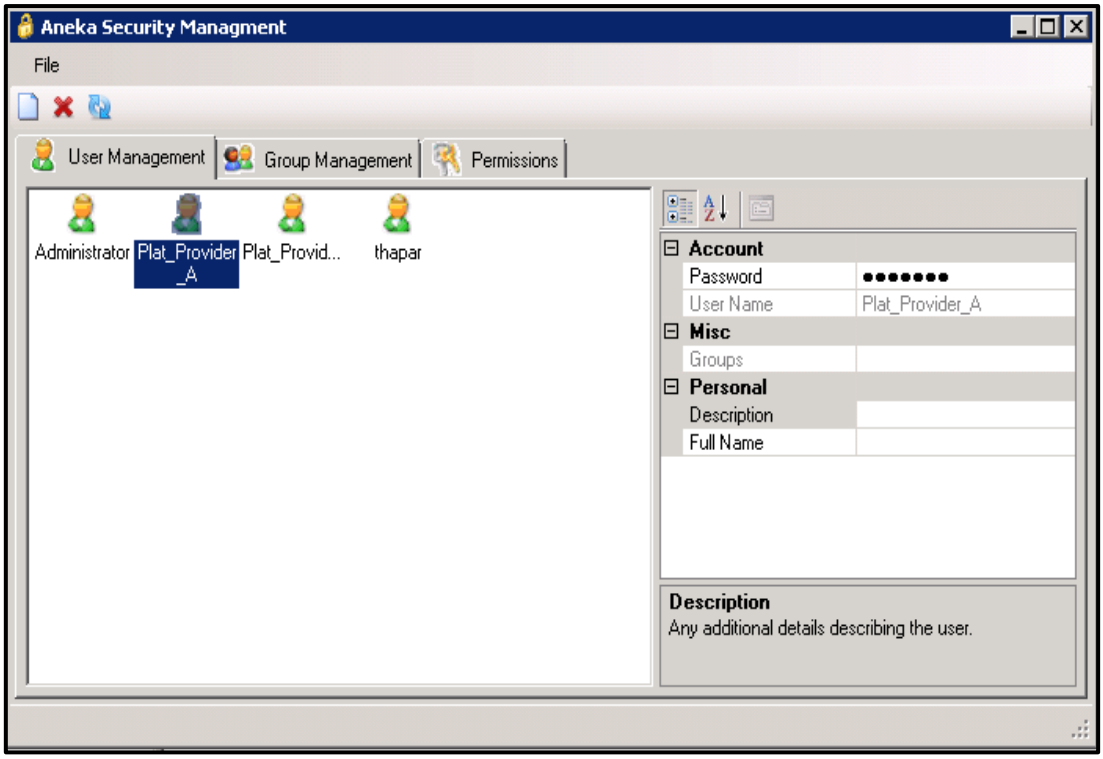

Figure 5.24: User Credentials in Aneka Setup

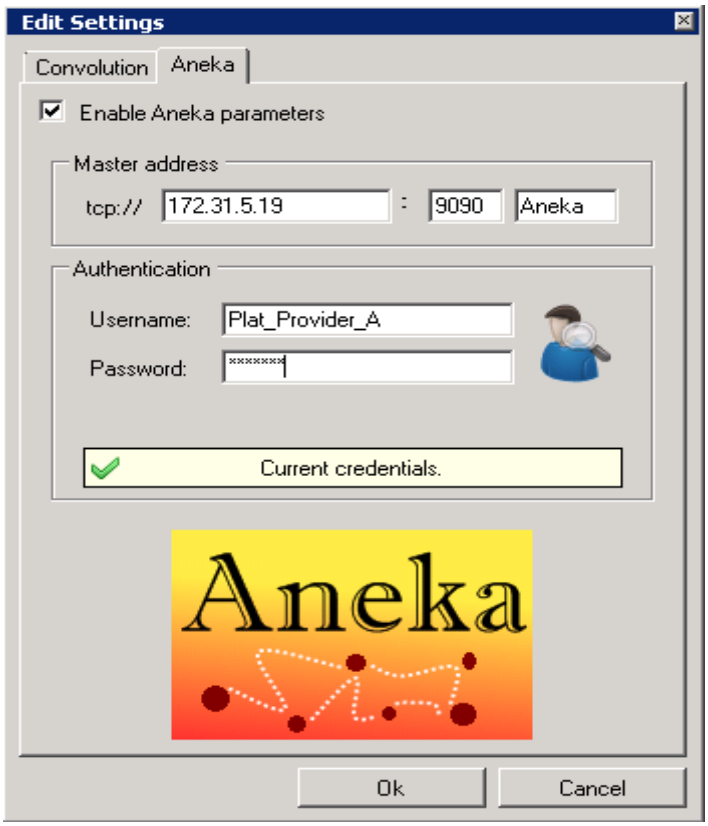

Figure 5.25: Convolution Connecting Credentials

5.3.6 Case I (User changes Infrastructure Provider):

As explained in above sections, user is using convolution application with Aneka as a platform to run that application. Now user wants to change Infrastructure provider. Again whole search processes are carried out and for implementation purpose say Infra_Provider_B is most capable. Infra_Provider_B uses Oracle Virtual Box as hypervisor but previous Infrastructure Provider uses VMWare Workstation 7.0. This section explains how virtual machine installed on one hypervisor can be ported to another hypervisor without loss of any data and configuration.

1. Convert virtual machine which carries user data and configuration into a virtual appliance. If user installed a virtual appliance then there is no need for this step. OVFTool is used to convert a virtual machine into a virtual appliance packaged into OVF format. Figure 5.26 shows how to convert a window xp virtual machine into a virtual appliance using the command prompt.

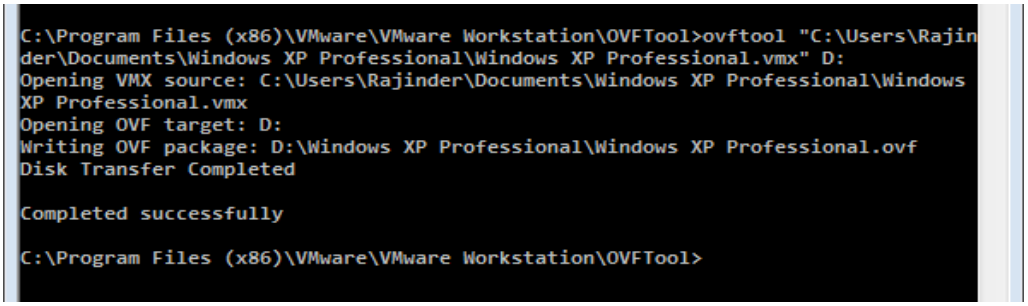

Figure 5.26: Using OVF Tool

2. After successfully completing the conversion process 2 files was created one is OVF file of virtual appliance and second is hard disk partition, shown in Figure 5.27.

Figure 5.27: Virtual Appliance of Virtual Machine

3. User gets space and all resource requirements on Infra_Provider_B and import virtual appliance in Oracle Virtual Box, as shown in Figure 5.28.

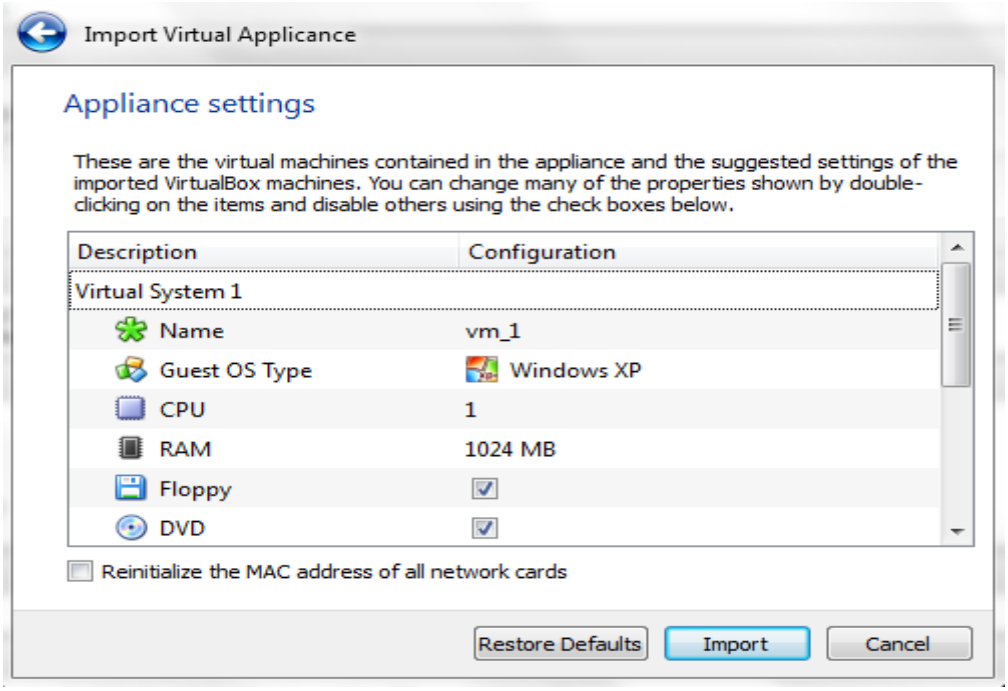

Figure 5.28: Virtual Box Import Appliance Window

4. After successful import of virtual appliance. User powered on imported virtual appliance all data and configurations are maintained.
5. User now just changes the ip of Infrastructure Provider in Aneka Interface and whole system is working as it was working before.

In this section steps are shown that how user can port virtual machine to heterogeneous hypervisors using virtual appliance and OVF packaging format.

5.3.7 Case II (User changes Platform Provider):

In this case two different platform providers providing same services or a third party platform provider rented its services to different brokers are needed. Taking the latter case, two different platform providers account have been created in Aneka as shown in Figure 5.24. Following steps shows how to change platform provider in this proposed architecture:

1. User disconnect with the previous provider.
2. User searches the capable provider with modified or same requirements. Search engine returns most capable provider using semantics and OWL as discussed in sections above.

3. A new username and password will be provided when user register with new platform provider. Using that username and password user will connect to platform provider as shown in Figure 5.25.
4. User will enter the IP address of its infrastructure provider so whole system will be running as it was before.

5.4 Architecture Validation

Buyya et. al. [35] proposed an architecture for automated IaaS management using virtual appliance. Proposed architecture is extended to PaaS. Some of key modifications in architecture are shown in Table 5.1 below.

Table 5.1 Difference between Architectures

	Buyya Architecture	**Proposed Architecture**
Service Layer	IaaS	PaaS
Matching Process	Static	Similarity Graph Based Matching
Appliance Search	Static	Ontology Based
User Data	With Provider	With Separate Third Party Provider
SLA	No Stated process	Sla Compensation Algorithm integrated with Architecture

5.5 Conclusion

This chapter demonstrates the Automated PaaS Management Architecture. Tools used for demonstration are explained briefly. Setup for Aneka 3.0 for proposed architecture is described. Conversion of virtual machine into a virtual appliance and how it is imported on different hypervisors is also shown. Next chapter concludes the thesis.

Chapter 6

Conclusion and Future Scope

This Chapter concludes the work presented in this thesis. At the end of this chapter, some future directions have been proposed which can be considered to improve the Automated PaaS Management architecture.

6.1 Conclusions

This thesis provides an insight to the essential aspect of standardization in Cloud computing, portability using virtualization and automation in cloud PaaS environment. Given the rapid uptake and great diversity of PaaS offerings, understanding portability at PaaS level is essential for supporting inter-cloud cooperation and seamless information exchange. So, to overcome many problems in Cloud PaaS an effective architecture for automated searching and deployment of virtual appliances is presented. This presented architecture includes four main improvements :

1. Automated searching of PaaS, IaaS provider and virtual appliance using semantic and similarity graph technologies while considering Service Level Agreement (SLA) and Quality of Service (QoS). This helps users to search providers not only on the basis of functional but also non-functional requirements.
2. Proposing an advertisement approach for PaaS and IaaS providers. So providers can easily lease their services. This helps small and middle level service providers to increase their business.
3. Converting user requirements to OVF to be a standard package format for cloud deployment. OVF helps user to change IaaS provider without changing PaaS provider and vice-versa. Infrastructure provider can be changed by just moving virtual appliance and deploy on new provider. For changing PaaS provider user just has to get new PaaS provider login and password.
4. Proper monitoring of Quality of Service by a third party SLA manager. Compensation mechanism for violation in signed SLA is managed by SLA manager.

One of the desirable attribute of our architecture is to allow users to present their requirements into high level and general software and hardware characteristics, which will be mapped to appliances and virtual units. Using appliances portability at Platform and Infrastructure Provider level can easily be achieved.

6.2 Thesis Contributions

Some of thesis contributions are:

I. Literature review is done in three different domains of Cloud Computing viz. Portability in Cloud, Resource Discovery in Cloud and Service Level Agreement in Cloud.
II. Developed a single framework to solve these three different issues of Cloud Computing.
III. A semantic based technique is developed for better resource discovery and matching process.
IV. A SLA violation compensation algorithm is presented.
V. Designed framework has been implemented using various softwares such as Asp.net, Protégé 4.0, Aneka 3.0 etc.

6.3 Future Scope

The proposed architecture can be further extended as :

1. A rating mechanism can be develop using SLA violation database so that user can get most rated provider.
2. A database converter can be developed to achieve portability at database level also.
3. Software used to implement different components of proposed architecture are developed by different companies so data is taken manually from one component to another. A broker can be developed which makes implementation automated.

References

[1] N. Sadashiv and D. Kumar, "Cluster, Grid and Cloud Computing: A Detailed Comparison," in *The 6th International Conference on Computer Science & Education (ICCSE 2011)*, SuperStar Virgo, Singapore, 2011, pp. 477-482.

[2] S. Kumar, *Cloud Computing- insight into New-Era Infrastructure*, 1st ed. India: Wiley India, 2011.

[3] L. Rodero-Merino, J. Caceres, M. Lindner and L. Vaquero, "A Break in the Clouds: Towards a Cloud Definition," *SIGCOMM Computing Communication Review*, vol. 39, no. 1, pp. 50-55, 2009.

[4] R. Buyya, C. S. Yeo, and S. Venogopal, "Market Oriented Cloud Computing: Vision, Hype and Reality for Delivering IT Services as Computing Utilities," in *10th IEEE International Conference on High Performance Computing and Communications, 2008. HPCC '08.*, Dalian, 2008, pp. 5-13.

[5] M. Armbrust et al., "Above the Cloud: A Berkeley view of Cloud Computing," University of California, Berkeley, USA, Techincal Report UCB/EECS-2009-28, 2009.

[6] I. Foster, Y. Zhao, I. Raicu, and S. LIU, "Cloud Computing and Grid Computing 360-Degree Compared," *IEEE Grid Computing Environments Workshop*, pp. 1-10, 2008.

[7] S. Pandey, W. Voorsluys, S. Niu, A. Khandokar, and R. Buyya, "An Autoomic Cloud Environment for Hosting ECG Data Analysis Services," in *Future Gemeration Computer Systems*, Amsterdam, 2012, pp. 147-154.

[8] R. Buyya, C. Vecchiola, and T. Selvi, *Mastering Cloud Computing*, 1st ed. New Delhi, India: McGraw Hill Education (India) Private Limited, 2013.

[9] K. Raghavendra, A. Akilan, N. Ravi, K. P. Kumar, and G. Varadan, "Satellite Data product Generation using Aneka Cloud," in *Research Demo at 10th IEEE International Symposium o Cluster, Cloud, and Grid Computing (CCGrid 2010)*, Melbourne, Australia, 2010.

[10] Google Inc. Google App Engine. [Online]. https://developers.google.com/appengine/docs/whatisgoogleappengine

[11] Google Inc. Google App Engine Pricing. [Online]. https://cloud.google.com/pricing/

[12] Dropbox Inc. Dropbox. [Online]. https://www.dropbox.com/

[13] E. Guizzo, "Robots with their heads in the cloud," *IEEE Spectrum*, 2011.

[14] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The Characteristics of Cloud Computing," in *39th International Conference on Parallel Processing Workshops*, San Diego, CA, 2010, pp. 275-279.

[15] E. Brown, F. Gillett, W. Saleh, and J. Staten. (2007) The Case For Virtual Appliances How Hypervisors Can Simplify Software Distribution. [Online]. www.vmware.com/go/vam_va_fd_forresterreport

[16] SalesForce. (2012) What is PaaS? [Online]. http://www.salesforce.com/paas/overview/

[17] I. Sriram and A. Khajeh-Hosseini, "Reseacrh Agenda in Cloud Technologies," University of Bristol, UK, Technical Report 2010.

[18] A. M. Vouk, "Cloud Computing Issues, Research, and Implementations," in *Proc. of the 30th International Conference of Information Technologies and Interfaces (ITI 2008)*, Cavtat/Dubrovnick, Croatia, 2008, pp. 31-40.

[19] G. Chockler, R. V. Renesse, and K. Birman, "Toward a Cloud Computing Research Agenda," *SIGACT News*, vol. 40, no. 2, pp. 68-80, 2009.

[20] P. Mell and T. Grance, "The NIST defination of Cloud Computing," NIST, USA, Techincal report 2011.

[21] J. McKendrick. (2010) Does Platform as a Service have interoperability issues? [Online]. http://www.zdnet.com/blog/service-oriented/does-platform-as-a-service-have-interoperability-issues/4890

[22] N. Loutas, E. Kamateri, F. Bosi, and K. Tarabains, "Cloud computing interoperability: the state of play," in *Third IEEE International Conference on Coud Computing Technology and Science*, 2011, pp. 752-757.

[23] R. H. Carpenter, "Walking From Cloud to Cloud: The Portability Issue in Cloud Computing," *Washington Journal of Law, Technology & Arts*, vol. 6, no. 1, pp. 1-14, Summer 2010.

[24] R. S. Pressman, "Software Engineering: A Practitioner's Approach". 7th Edition: McGraw Hill International, ch. 14, pp. 398-415.

[25] M. N. Shirazi, H. C. Kauan, and H. Dolatabadi, "Design Patterns to Enable Data Portability between Clouds' Databases," in *12th International Conference on*

Computational Science and Its Applications, 2012, pp. 117-120.

[26] H. Sakai, "Standardization Activities for Cloud Computing," NTT Information Sharing Platform Laboratories, Musashino-shi, Japan, Technical Review 2011.

[27] DMTF, "Open Virtualization Format v.1.0," DMTF, White Paper DSP 2017 v1.0.0, 2007.

[28] DMTF Inc., "Enabling Portability & Simplified Deployment of Virtual Appliances," Technical Note 2008.

[29] Vmware Inc., "Developer's Guide to Building vApps and Virtual Appliances," Palo Alto, CA, User Manual EN-000831-00, 2012.

[30] C. Sapuntzakis et al., "Virtual Appliance for deploying and Maintaining Software," in *The 17th USENIX Conf on System Administration*, 2003, pp. 181-194.

[31] DMTF Inc. (2012) DMTF's VMAN Standard Achieves ANSI Adoption. [Online]. http://www.dmtf.org/content/dmtfs-vman-standard-achieves-ansi-adoption

[32] A. Govindarajan and G. Lakshmanan, "Overview of Cloud Standards," *Cloud Computing, Springer London*, vol. 1, no. 1, pp. 77-89, 2010.

[33] Z. Hill and M. Humphery, "CSAL: A Cloud Storage Abstraction Layer to Enable Portable Cloud Applications," in *IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, pp. 504-511.

[34] F. Galán, M. Gómez, F. de la Iglesia, I. Blasco, and D. Morán, "Autoconfiguration of Enterprise-class Application Deployment in Virtualized Infrastructure Using OVF Activation Mechanisms," in *6th International DMTF workshop on Systems and Virtualization Management (SVM 2012)*, 2012, pp. 412-421.

[35] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, "An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery," in *IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, Australia, 2010, pp. 104-112.

[36] M. Hogen, F. Liu, A. Sokol, and J. Tong, "Cloud Computing Standards Roadmap," NIST, Technical Report 500-291,.

[37] S. Ortiz, "The Problem with Cloud-Computing Standardization," *Computer*, vol. 44, no. 7, pp. 13-16, July 2011.

[38] NIST-SAJACC. Standards Acceleration to Jumpstart Adoption of Cloud Computing. [Online]. http://collaborate.nist.gov/twiki-Cloud-Computing/bin/view/CloudComputing/SAJACC

[39] SNIA. (2009) Cloud Storage Initiative. [Online]. http://snia.org/forums/csi

[40] D. L. Liu, and H. Schmeck Ding, "Service Discovery in Self-organizing Service-oriented Environments," in *Asia-Pacific Services Computing Conference*, Hangzhou, China, 2010, pp. 717-724.

[41] L. D. Ngan and R. Kanagasabai, "OWL-S Based Semantic Cloud Service Broker," in *2012 IEEE 19th International Conference on Web Services*, 2012, pp. 560-567.

[42] R. Raman, M. Livny, and M. Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing," in *In: Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, Chicago, IL, USA, 1998.

[43] M. Horridge, "A Practical Guide To Building OWL Ontologies," The University Of Manchester, Manchester, Tutorial Guide Edition 1.3, 2011.

[44] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, vol. 43, no. 4-5, pp. 907-928, November 1995.

[45] C. Shrinky. Clay Shirky's Writings About the Internet. [Online]. http://www.shirky.com/writings/ontology_overrated.html

[46] T. Andreasen, H. Bulskov, and R. Knappe, "From Ontology over Similarity to Query Evaluation," in *2nd International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems (ODBASE)*, Catania,Sicily,Italy, November 2003.

[47] T. Andreasen, A. Motro, H. Christiansen, and H. L. Larsen, "On Measuring Similarity for Conceptual Querying," in *5th International Conference on Flexible Query Answering Systems*, Copenhagen, Denmark, 2002, pp. 27-29.

[48] K. M. Sim, "Agent-Based Cloud Computing," *Services Computing, IEEE Transactions on*, vol. 5, no. 4, pp. 564-577, October 2011.

[49] M. Godse and S. Mulik, "An Approach for Selecting Software-as-a-Service (SaaS) Product," in *IEEE International Conference on Cloud*, Bangalore, India, 2009, pp. 155-158.

[50] J. Kang and K. M. Sim, "Towards Agents and Ontology for Cloud Service Discovery," in *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2010, pp. 483-490.

[51] P. Bianco, G. A. Lewis, and P. F. Merson, "Service Level Agreements in Service-Oriented Architecture Environments," Carnegie Mellon University, Techincal Note CMU/SEI-2008-TN-021 , 2008.

[52] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services," *J. Network Syst. Manage*, vol. 11, no. 1, pp. 57-81, 2003.

[53] L. Mastroeni and M. Naldi, "Violation of Service Availability Targets in Service Level Agreements," in *Proceedings of the Federated Conference on Computer Science and Information Systems*, 2011, pp. 537-540.

[54] G. Yonggen, Z. Wei, and T. Jie, "A Study of SLA Violation Compensation Mechanism in Complex Cloud Computing Environment," in *Second International Conference on Instrumentation & Measurement, Computer, Communication and Control*, 2012, pp. 1448-1451.

[55] I. Brandic, V. C. Emeakaroha, S. Acs, A. Kertesz, and G. Kecskemeti, "LAYSI: A Layered Approach for SLA-Violation Propagation in Self-manageable Cloud Infrastructures," in *34th Annual IEEE Computer Software and Applications Conference Workshops*, 2010, pp. 365-370.

[56] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Laguage User Guide*, 2nd ed. New Delhi, India: Pearson Education, Inc., 2007.

[57] Vmware Inc. Vmware Studio. [Online]. http://www.vmware.com/appliances/learn/vmware_studio.html

[58] Vmware Inc. (2011) Vmware Appliances Market Place. [Online]. https://solutionexchange.vmware.com/store/category_groups/19

[59] Citrix. (2010) Project Kensho v1.1 Technology Preview. [Online]. http://community.citrix.com/disply/xs/kensho

[60] Vmware Inc. (2013) Vmware OVF Tool. [Online]. http://www.vmware.com/resources/techresources/1013

[61] Protege. (2013) Protege Ontology Editor and Knowledge Acquisition System. [Online]. http://protege.stanford.edu/

[62] Vmware Inc., "OVF Tool User Guide," Palo Alto, CA, User Manual EN-000143-

00, 2009.

[63] ManjraSoft Pvt. Ltd., "Installation and User Guide Aneka 3.0," University of Melbourne, Australia, Melbourne, User Manual 2012.

[64] ManjraSoft Pvt. Ltd. [Online]. http://www.manjrasoft.com

List of Publications

1. Rajinder Sandhu and Inderveer Chana, "Securing Virtual machines in Cloud Environment using OVF", in *Second International Conference on Advances In Electronics, Electrical And Computer Engineering*, Dherahun, India, 2013.