Cloud Service Model Patterns

KEIKO HASHIZUME, Florida Atlantic University
EDUARDO B. FERNANDEZ, Florida Atlantic University
MARIA M. LARRONDO-PETRIE EDUARDO B. FERNANDEZ, Florida Atlantic University

In this paper, we develop two patterns for two of the cloud service models: Infrastructure-as-a-Service and Platform-as-a-Service. We assume that our audience will include cloud system designers as well as cloud application builders.

Categories and Subject Descriptors: **D.2.11 [Software Architectures]: Patterns.**

General Terms: Security

Additional Key Words and Phrases: Cloud models, cloud services, Infrastructure-as-a-service, Platform-as-a-service, pattern

1. INTRODUCTION

Cloud computing is a new paradigm that improves the utilization of resources and decreases the power consumption of hardware. Cloud computing allows users to have access to resources, software, and information using any device that has access to the Internet. The users consume these resources and pay only for the resources they use.

A cloud model provides three types of services (Mather el al. 2009): Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). IaaS provides processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. PaaS offers platform layer resources, including operating system support and software development frameworks to build, deploy and deliver applications into the cloud. SaaS provides end-user applications that are running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email).

In this paper, we develop two patterns for two of the cloud service models: Infrastructure-as-a-Service and Platform-as-a-Service. We assume that our audience will include cloud system designers as well as cloud application builders. Section 2 presents the Cloud Infrastructure pattern. Next, in Section 3 we present the Platform-as-a-Service pattern. In Section 4, we present some conclusions and possible future work.

2. INFRASTRUCTURE-AS-A-SERVICE

2.1 Intent

The Infrastructure-as-a-Service describes the infrastructure to allow the sharing of distributed virtualized computational resources such as servers, storage, and network.

2.2 Context

Distributed systems where we want to improve the utilization of resources and provide convenient access to all users.

2.3 Problem

Some organizations do not have the resources to invest in infrastructure, middleware, or applications needed to run their businesses. Also, they may not be able to handle higher demands, or they cannot afford to maintain and store unused resources. How can they get access to computational resources?

Forces

- *Transparency* - The underlying architecture should be transparent to its users. Users should be able to use the provider's services without understanding its infrastructure.
- *Flexibility* - Different infrastructure configurations and amounts of resources can be demanded by users.
- *Elasticity* - Users should be able to expand or reduce resources in order to meet the different needs of their applications.
- *Pay-per-use* - Users should only pay for the resources they consume.
- On-demand-service - Services should be provided on demand.
- *Manageability* - In order to manage a large amount of service requests, the cloud resources must be easy to deploy and manage.
- *Accessibility* - Users should access resources from anywhere at anytime.
- *Testability* - We intend to develop system programs in this environment and we need to test them conveniently.
- *Shared resources* - Many users should be able to share resources in order to increase the amount of resource utilization and thus reduce costs.
- *Isolation* - Different user execution instances should be isolated from each other.
- *Shared Non-functional requirements provision (NFRs)* - Sharing of the costs to provide NFRs is necessary to allow providers to offer a higher level of NFRs.

2.4 Solution

The solution to this problem is a structure that is composed of many servers, storage, and a network, which can be shared by multiple users and accessible through the Internet. These resources are provided to the users as a form of service called Infrastructure-as-a-Service (IaaS). IaaS is based on virtualization technology which creates unified resources that can be shared by different applications. This foundation layer – IaaS – can be used as a reference for non-functional requirements.

Structure

Figure 1 shows a class diagram for a cloud infrastructure. The **Cloud Controller** is the main component which processes requests from a **Party**. A Party can be an institution or a user (customers and administrators). A Party can have one or more **Accounts**. The Cloud Controller coordinates a collection of services such as VM scheduling, authentication, VM monitoring and management. When a Cloud Controller receives a request from the party to create a VM, it requests its corresponding **Cluster Controllers** to provide a list their free resources. With this information, the Cloud Controller can choose which cluster will host the requested virtual machine. A Cluster Controller is composed of a collection of **Node Controllers**, which consist of a pool of Servers that host **Virtual Machine (VM)** instances. The Cluster Controller handles the state information of its Node Controllers, and schedules incoming requests to run instances. A Node Controller controls the execution, monitoring, and termination of the VMs through a **Virtual Machine Monitor (VMM)** which is the one responsible to run VM instances. The Cloud Controller retrieves and stores user data and **Virtual Machine Images (VMI)**. The **Virtual Machine Image Repository** contains a collection of Virtual Machine Images that are used to instantiate a VM. The **Dynamic Host Configuration Protocol (DHCP)** server assigns a MAC/IP (Media Access Control/Internet Protocol) pair address for each VM through the Cloud Controller, and requests the **Domain Name System (DNS)** server to translate domain names into IP addresses in order to locate cloud resources.

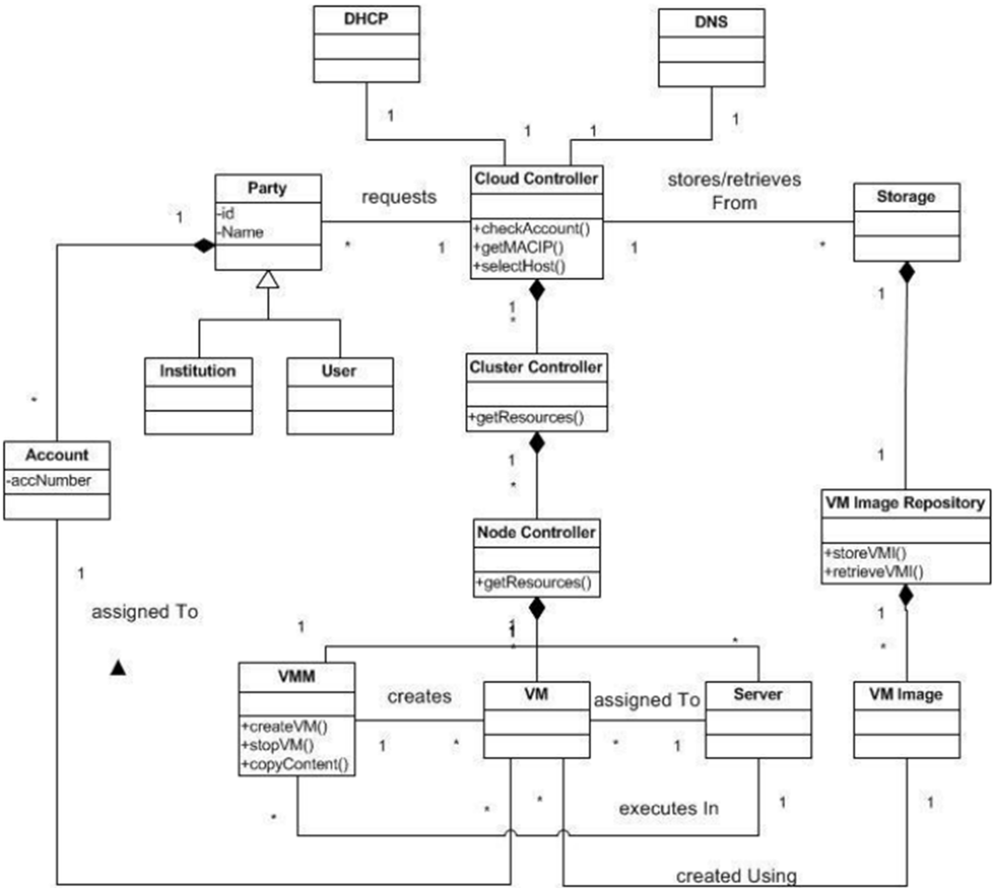

Fig. 1. Class Diagram for a Cloud Infrastructure

Dynamics

Some use cases are the following (NIST):

- Open/close an account (actor: user)
- Copy data objects into/out a cloud (actor: administrator)
- Erase data objects in a cloud (actor: administrator)
- Store/Remove VM images (actor: administrator, user)
- Create a VM (actor: user)
- Migrate a VM (actor: administrator, user)

We show two UCs below:

UC1: Create a Virtual Machine (Figure 2)
Summary: Create of a Virtual Machine for a party, assign to it requested resources and assign it to a server.
Actor: Party
Precondition: The party has a valid account
Description:
(a) A party requests a VM with some computational resources to the Cloud Controller.
(b) The Cloud Controller verifies whether the requester has a valid account.

(c) The Cloud Controller requests the available resources to the Cluster Controllers closer to the location of the party. In turn, the Cluster Controller queries its Node Controllers about their available resources. In the sequence diagram, there is only one Cluster Controller and one Node Controller in order to maintain the diagram simple, but there can be more Clusters and Node Controllers.
(d) The Node Controller sends the list of its available resources to the Cluster Controller, and the Cluster Controller sends it back to the Cloud Controller.
(e) The Cloud Controller chooses the first Cluster Controller that can support the computational resources requirements.
(f) The Cloud Controller requests a MAC/IP pair address from the DHCP server for the new VM.
(g) The Cloud Controller retrieves a virtual machine image from the repository (VMI)
(h) The Cloud Controller sends a request to the Cluster Controller to instantiate a VM.
(i) The Cluster Controller forwards the request to the Node Controller which forwards it to the VMM.
(j) The VMM creates a VM with the requested resources.
(k) The VMM assigns the VM to one of the servers.

Postcondition: A virtual machine is created and assigned to an account and a server.

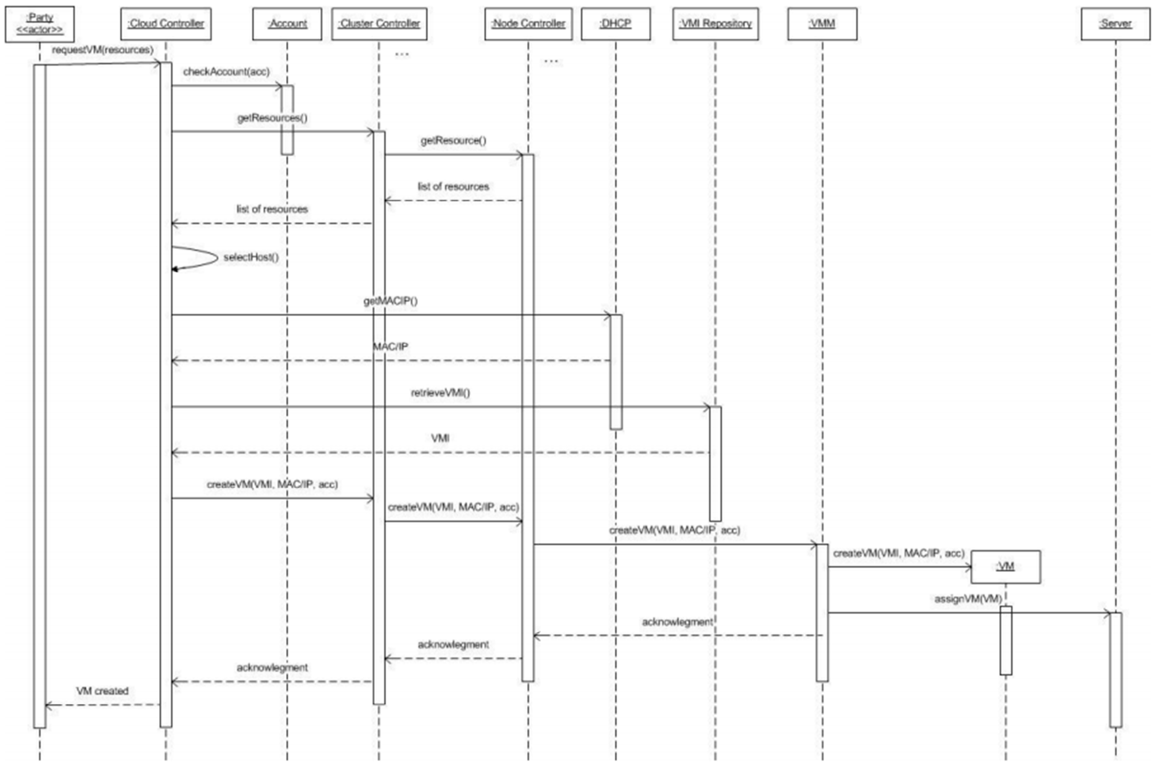

Fig. 2. Sequence Diagram for Use Case Create a Virtual Machine

UC2: Migrate a Virtual Machine (Figure 3)

The administrator can migrate a VM to a specific Node Controller that can be located in the same or in a different Cluster Controller. The administrator can also migrate a VM to a specific location or to the first node that has the available resources. For the scenario below, we assume that the administrator will move a VM to the first available Node Controller within the same Cluster Controller.

Summary: A VM is migrated from one Node Controller to another one.

Actor: Administrator

Precondition: a VM resides in some Node Controller (Compute Node Source)

Description:

(a) The Administrator requests to the Cloud Controller to migrate a VM. However, the migration process can be automatic due to load balancing for example.
(b) The Cloud Controller sends a request to the Cluster Controller to start the migration of the VM.
(c) The Cluster Controller requests the Node Controller Source to stop the VM. The Node Controller Source forwards this request to the VMM Source.
(d) The VMM Source stops the VM and copies the content of the VM.
(e) Do the same as UC1
(f) The VMM Source sends the content of the VM to the VMM destination.
(g) The VMM destination copies the content into the new VM.

Postcondition: The VM has migrated to another host

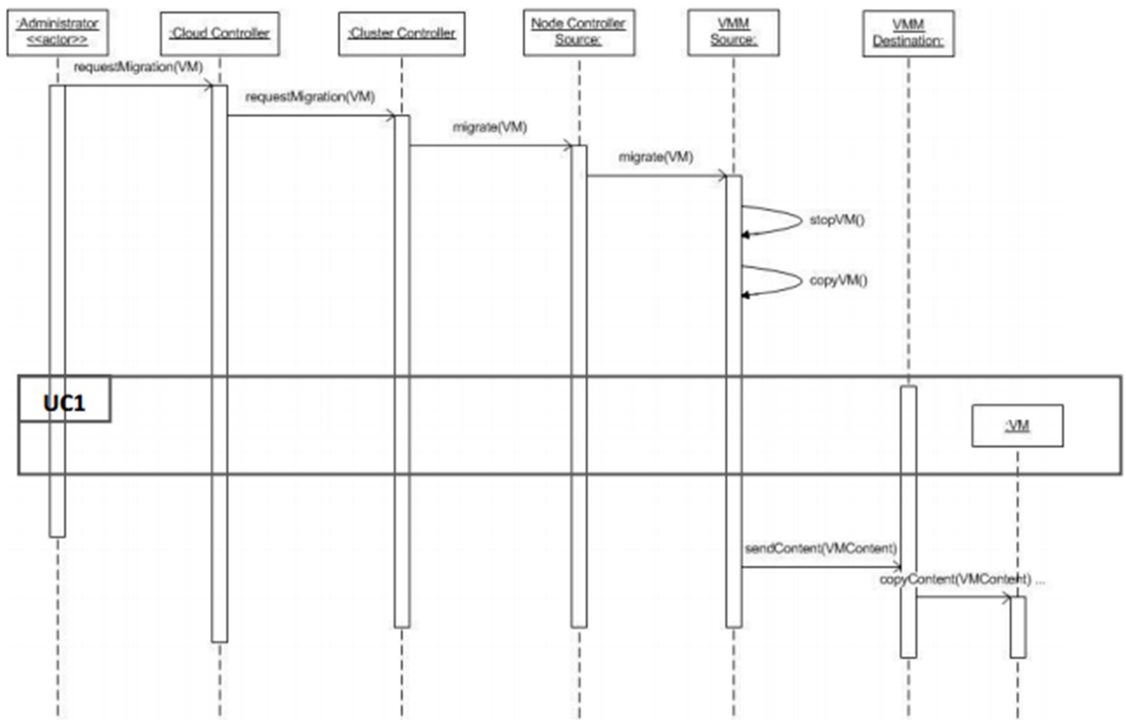

Fig. 3. Sequence Diagram for Use Case Migrate a Virtual Machine.

2.5 Implementation

Eucalyptus (Eucalyptus) is an open source software that allows to implement Infrastructure as a Service in order to run and control virtual machine instances via Xen and KVM. Eucalyptus consists of five main components that are described in Figure 4 (Baun and Kunze 2009).

The two higher level components are: the Cloud Controller and Walrus. The Cloud Controller is a Java program that offers EC2-compatible SOAP and web interfaces. Walrus is a data storage where users can store and access virtual machine images and their data. Walrus can be accessed through S3-compatible SOAP and REST interfaces. Top-level components can aggregate resources from several clusters. Each cluster needs a Cluster Controller which is typically deployed on the head-node of a cluster. Each node will also need a Node Controller for controlling the hypervisor. Cluster Controller and Node Controllers are deployed as web services, and communications between them takes place over SOAP with WS-Security (Hashizume and Fernadez 2009).

A cloud can be setup as a single-cluster where the Cloud Controller and the Cluster Controller are located on the same machine, which are referred to front-end. All other machines running the Node Controllers are referred as back-end. However, there could be also a more advanced configuration which comprises several Cluster Controller or Walrus deployed in different machines.

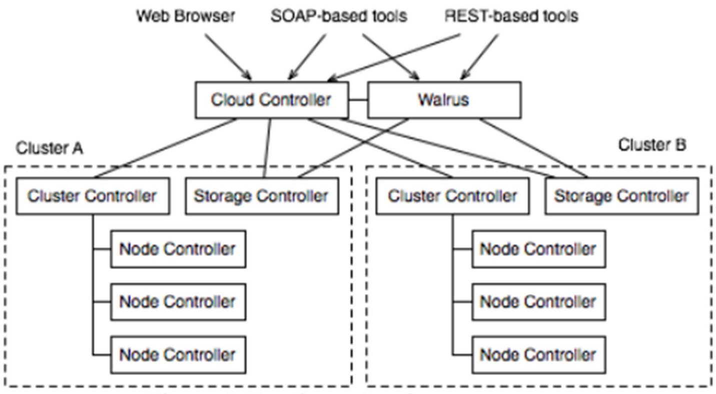

Fig. 4. Eucalyptus' main components.

A typical configuration includes (Ubuntu):

- 1 Cloud Controller (CPU-1GHz, Memory-512MB, Disk-5400rpm IDE, Disk space-40GB)
- 1 Walrus Controller (CPU-1GHz, Memory-512MB, Disk-5400rpm IDE, Disk space-40GB)
- 1 Cluster Controller + Storage Controller (CPU-1GHz, Memory-512MB, Disk-5400rpm IDE, Disk space-40GB)
- Nodes (VT extensions, Memory-1GB, Disk-5400rpm IDE, Disk space-40GB)

2.6 Known Uses

Eucalyptus (Eucalyptus) is an open source framework used for hybrid and private cloud computing.

- OpenNebula (OpenNebula) is an open source toolkit to build clouds.
- Nimbus (Nimbus) is an open source set of tools that offers IaaS capabilities to the scientific community.
- Amazon's EC2 (Amazon) provides compute capacity though web services.
- HP Cloud Services (HP) is a public cloud solution that provides scalable virtual servers on demand.
- IBM SmartCloud Foundation (IBM) offers servers, storage and virtualization components in order to build private, public and hybrid clouds.

2.7 Consequences

The Cloud Infrastructure Pattern provides the following benefits:

- *Transparency* - Cloud users are usually not aware where their virtual machines are running or where their data is stored. However, in some cases users can request a general location zone for virtual machines or data.
- *Flexibility* - Cloud users can request different types of computational and storage resources. For instance, Amazon's EC2 (Amazon) provides a variety of instance types and operating systems.
- *Elasticity* - Resources provided to users can be scaled up or down depending on their needs. Multiple virtual machines can be initiated and stopped in order to handle increased or decreased workloads.
- *Pay-per-Use* - Cloud users can save on hardware investment because they do not need to purchase more servers. They just need to pay for the services that they use. Cloud services are usually charged using a fee-for-service billing model (Centre 2010). For instance, users pay for the storage, bandwidth or computing resources they consume per month.
- *On-demand-services* - IaaS providers deliver computational resources, storage and network as services with one click.
- *Manageability* - Users place their requests to the cloud administrator who allocates, migrates, and monitors VMs.

- *Accessibility* - Cloud services are delivered using user-centric interfaces via the Internet (Wang et al. 2008) from anywhere and anytime.
- *Testability* - Having an isolated environment allows testing system programs without affecting the execution of other virtual machines.
- *Shared resources* - Virtualization enables to share a pool of resources such as processing capacity, storage, and networks. Thus, higher utilization rate can be reached (Amrheim).
- *Isolation* - A VMM provides strong isolation between different virtual machines, whose guest operating systems are then protected from one another (Karger and Safford 2008).
- *Shared Non-functional requirements (NFRs)* provision - Some IaaS providers offer security features such as authentication and authorization to customers that can be added as part of the service. Sharing allows the provider to offer a higher degree of NFRs at a reasonable cost.

The Cloud Infrastructure Pattern has the following liabilities:

- Cloud computing is dependent on network connections. While using cloud services, users must be connected to the Internet, although a limited amount of work can be done offline.
- The Cloud may bring security risks associated to privacy and confidentiality areas since the users do not have the control of the underlying infrastructure.
- The isolation between VMs may not be so strong (Hashizume et al. 2012).
- Virtualization increases some performance overhead.

2.8 Related Patterns

The Virtual Machine Operating System (Fernandez and Sorgente 2005)) describes the VMM and its created VMs from the point of view of an OS architecture.

- The Grid Architectural Pattern (Camargo et al. 2006) allows the sharing of distributed and heterogeneous computational resources such as CPU, memory, and disk storage for a grid environment.
- Misuse patterns (Hashizume et al. 2012) describe possible attacks to cloud infrastructures.
- The Platform-as-a-Service (PaaS) pattern describes development platforms that provide virtual environments for developing applications in the cloud.
- Party (Fowler 1997) indicates that users can be individuals or institutions.

3. PLATFORM-AS-A-SERVICE

3.1 Intent

The Platform-as-a-Service (Pass) provides virtual environments for developing, deploying, monitoring, and managing applications online without the cost of buying and managing the corresponding software tools or hardware.

3.2 Content

PaaS services are built on top of the cloud's Infrastructure-as-a-Service (IaaS), which provides the underlying infrastructure.

3.3 Problem

Organizations may want to develop their own custom applications without buying and maintaining the developing tools, databases, operating systems, and infrastructure underneath them. Also, when the team is spread across several places, it is necessary to have a convenient way to coordinate their work. How do we provide these functions?

This problem is affected by the following forces:

- *Collaboration* - sometimes teams of developers are located in different geographic locations. When working on a project, they all should have access to the development tools, code, and data.
- *Coordination* - When many developers work on a complex project, we need to coordinate their work.

- *Elasticity* - There should be a way to increase or decrease resources for more compute-intense development and deployment tasks.
- *Pay-for-use* - Party should only pay for the resources that they use.
- *Transparency* - Developers should not be concerned about the underlying infrastructure including hardware and operating systems, and its configuration for development and deployment.
- *On-demand services* - The developers should have the ability to request for an application tool and start using it.
- *Accessibility* - Developers should be able to access the tools via standard networks and from anywhere at any time.
- *Testability* - We intend to develop application programs in this environment and we need to test them conveniently.
- *Versatility* - The platform should be able to be used to build applications for any domain or type of application. Also, different options for developing tools should be offered to the users.
- *Simplification* - Developers should be able to build applications without installing any tool or specialized software in their computers.

3.4 Solution

PaaS offers virtual execution environments with shared tools and libraries for application development and deployment into the cloud. PaaS uses IaaS as a foundation layer (servers, storage, and network). PaaS hides the complexity of managing the infrastructure underneath.

Structure

Figure 5 shows a class diagram for a cloud Platform-as-a-Service (PaaS). The **PaaS Provider** processes requests from **Parties**. A Party can be an institution or a user (developers, administrators). The Party will choose the developing tools from the **Software Repository** which contains a list of developing tools available for the users. The PaaS provider offers **Virtual Environments** such as **Development Environment** and **Deployment Environment**. The Development Environment is composed of **Development Tools**, **Libraries**, **Databases**. The Virtual Environments are built upon the **IaaS** (Infrastructure-as-a-Service) which provides the underlying hardware. The same PaaS Provider can manage the IaaS, or it can be managed by a third-party service provider.

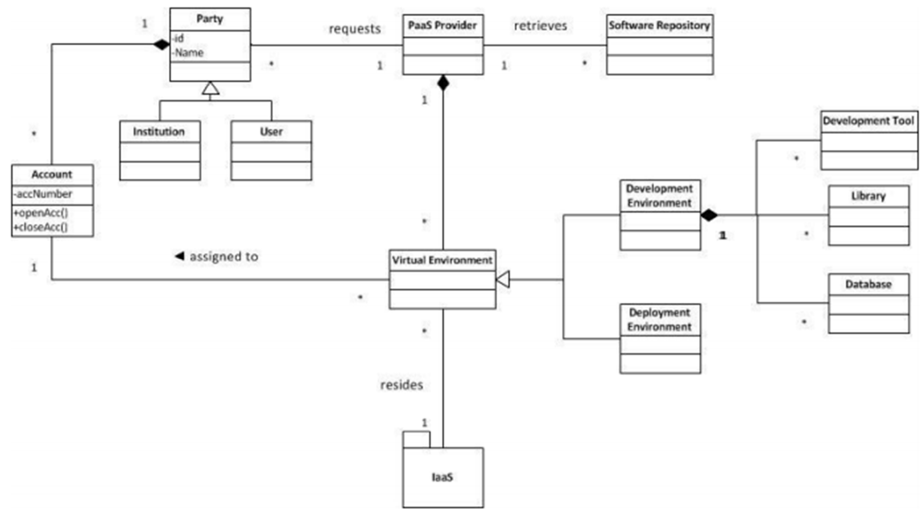

Fig. 5. Class Diagram for PaaS Pattern.

Dynamics

Some use cases are the following (Dosani 2010):

- Open/close an account
- Request a virtual environment
- Use a virtual environment
- Install development software
- Deploy application
- Undo deploy application

UC1: Consume Development Software (Figure 6)
Summary: A party requests to use a development application for the first time
Actor: Party
Precondition: The Party has an account
Description:
(a) The Party requests to use a particular development software
(b) The PaaS Provider checks if the Party has a valid account.
(c) The Party downloads the client applications on its machine
Postcondition: The client application is download in the party's machine

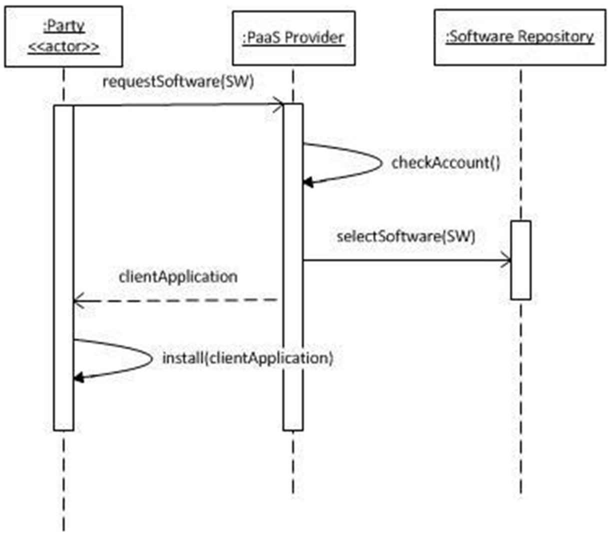

Fig. 6. Sequence Diagram for Consuming Development Software.

UC2: Deploy an Application (Figure 7)
Summary: A party requests to deploy his application into the cloud, so the application can be accessed by end-users from anywhere at any time.
Actor: Party (developer)
Precondition: The party has an account
Description:
(a) A party requests to deploy his application into the cloud.
(b) The PaaS Provider checks if the party has a valid account.
(c) The PaaS Provider calculates the computational resources needed for the deployment such as number of virtual machines.
(d) The PaaS Provider requests to the IaaS to create a set of Virtual Machines
(e) The PaaS Provider installs and runs the code
Postcondition: The application is running and ready to be accessed by the end-users

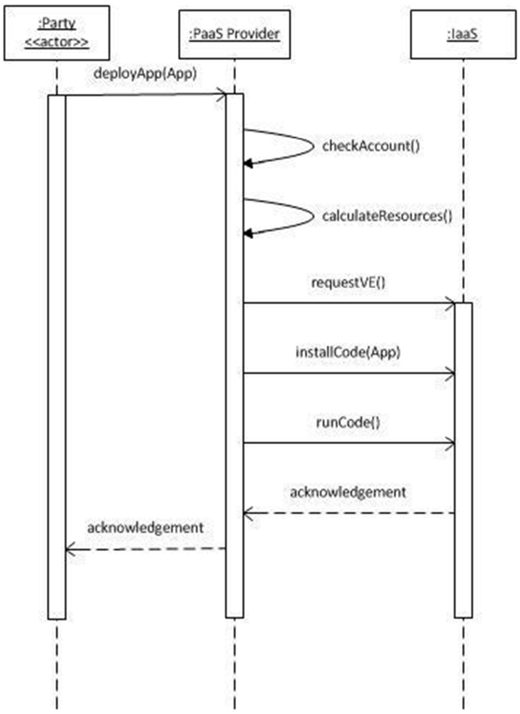

Fig. 7. Sequence Diagram for Deploying an Application.

3.5 Implementation

As an example of implementation of a typical PaaS approach we present here the approach used by Force.com. Force.com (Salesforce 1) is a cloud Platform-as-a-Service system from Salesforce.com. Force.com's platform provides PaaS services as a stack of technologies and services covering from infrastructure, database as a service, integration as a service, logic as a service, user interface as a service, development as a service, and AppExchange (Salesforce 2) in order to create of business applications. Figure 8 shows the stack of Force.com's technologies and services, and it includes:

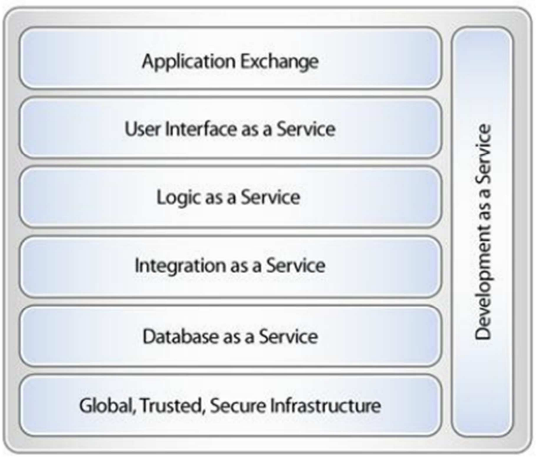

Fig. 8. The Force.com stack and services (from (Salesforce 2)).

Infrastructure - The foundation of Force.com platform is the infrastructure that supports the other layers. Force.com uses of three geographically dispersed data centers and a production-class development lab which use replication to mirror the data at each location.
Database as a service - It enables customers to create customized data objects, such as relational tables, and use metadata to describes those objects. Force.com provides data security by providing features such as user authentication, administrative permissions, object-level permissions, and field-level permissions.
Integration as a service - Force.com provides integration technologies that are compliant with open Web services and service-oriented architecture (SOA) standards, including SOAP, WSDL, and WS-I Basic Profile (Fernandez et al. 2010). Force.com offers different prepackaged integration solutions such as Web Services API, Web Services Apex, callouts and mashups, and outbound messaging.
Logic as a service - Force.com provides three options for implementing an application's business processing logic: declarative logic (unique fields, audit history tracking, history tracking, and approval processes), formula-based logic (formula fields, data validation rules, workflow rules, and approval processes), and procedural logic (Apex triggers and classes).
User Interface as a Service - Force.com provides two types of tools for creating the user interface of applications built on the platform applications: Force.com's Builder and Visualforce. Builder creates metadata, which Forces.com uses to generate a default user interface for each database object with its corresponding methods such as create, edit, and delete. With Visualforece, developers can use standard web development technologies such as HTML, Ajax, and Adobe Flex to create user interfaces for their cloud applications.
Development as a Service - Force.com offers some features to create cloud applications: Metadata API, Integrated Development Environment (IDE), Force.com Sandbox, and Code Share. Metadata API allows modifying the XML files that control an organization's metadata. The IDE provides a code editor for adding, modifying and testing Apex applications. Apex is the Force.com proprietary programming language. Also, multiple developers can share a code source repository using the synchronization features of the IDE. Force.com Sandbox provides a separate cloud-based application environment for development, quality assurance, and training. Force.com Code Share allows developers from different organizations to collaborate on the development, testing, and deployment of cloud applications.
Force.com IDE (Salesforce 3) is a client application for creating, modifying, and deploying applications. Once the user downloads the IDE in his local machine, he can start coding. The IDE is in communication with the Force.com platform servers. There are two types of operations: online and offline. For example, in the online mode, when a class is saved, the IDE sends the class to the Force.com servers that compile the class and return any result (error message). In the offline mode, all changes you performed on a local machine, and once you connect to Force.com, all those changes are submitted and committed. Force.com provides a built-in support for automated testing. Once an application is developed in the development environment, it may be migrated to another environment such as testing, or production. The Force.com IDE provides an easy way to deploy; just right click on the component and select Deploy to Server.
Application Exchange - AppExchange is a cloud application marketplace where users can find applications which are delivered by partners or third-party developers.

Force.com offers environments (Salesforce 4) where users can start developing, testing, and deploying cloud computing applications. There are different types of environments such as Production, Development, and Test Environments. The Production environment stores live data, while the Development Environment stores test data and are used for developing and testing applications. The Development Environment has two types: Developer Edition and Sandbox. Sandbox is a copy of the production environment that can include data, configurations, or both. A Developer Edition environment (Salesforce 5) includes the following developer technologies: Apex programming language, Visualforce for building custom user interface and controllers, the Integration APIs, and more. Figure 9 depicts the platform for the Developer Edition environment. The Force.com's virtual environments run on Salesforce's infrastructure.

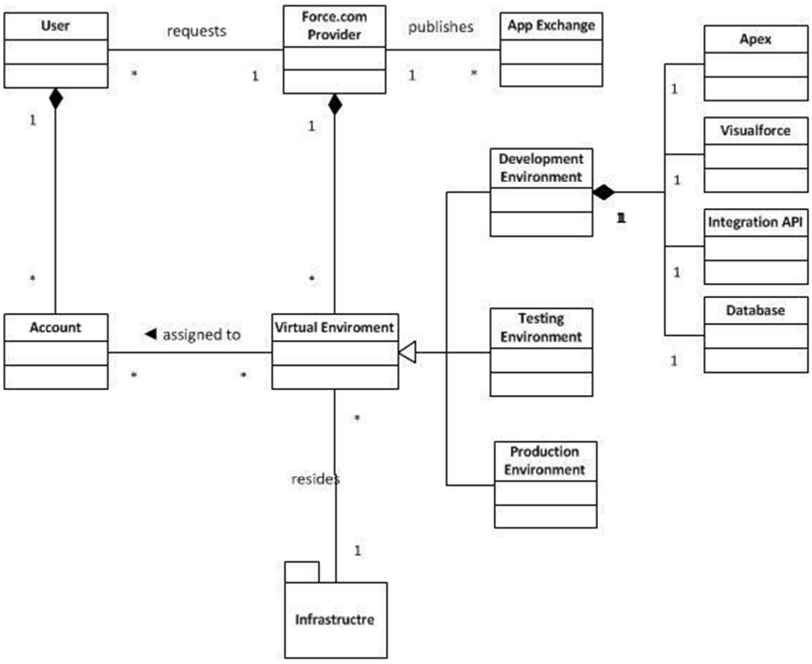

Fig. 9. Class Diagram of Force.com's PaaS architecture.

Force.com uses different security techniques to defense its platform from different types of threats (Salesforce 6).

- User authentication: Most users are authenticated on the login page, but also there are other forms of user authentication: delegated authentication, and Security Assertion Markup Language (SAML).
- An authenticated session needs to be established before accessing the Force.com SOAP API and Metadata API.
- Force.com secure its network using different mechanisms such as Stateful packet inspection (SPI), bastion hosts, two-factor authentication processes, and end-to-end TLS/SSL cryptographic protocols.
- For sensitive data such as customer passwords, Force.com applies an MD5 one-way cryptographic hash function, and supports encryption of field data.
- At an infrastructure and network level, salesforce.com applies rigorous security standards, such as SysTrust SAS 70 Type II.
- Salesforce.com implements industry best practices to harden the host computers. For example, all hosts use Linux or Solaris distribution with non-default configurations and minimal processes, user accounts, and network protocols.

3.6 Know Uses

Google App Engine (Google) provides an environment to build and host web applications on Google's infrastructure. Google App Engine supports two application environments: Java and Python.

- Microsoft Azure (Microsoft) provides a platform to build, deploy, and manage applications. It provides different programming languages such as .Net, Java, PHP and others in order to build applications.
- Salesforce [21] offers a development platform to build custom applications. (See Implementation)
- IBM SmartCloud Applications Services (Dosani 2010) delivers a collaborative environment that supports the full lifecycle for software development, deployment, and delivery.

3.7 Consequences

The PaaS pattern provides the following benefits:

- *Collaboration* - Geographically dispersed developers can collaborate on the same project because the code is managed online (Lawton 2008).
- *Coordination* - A project can be conveniently administered from a central point.
- *Elasticity* - The resources (storage, networking resources, and servers) needed to develop and deploy an application can grow or shrink to accommodate varying workload volumes. Scaling application deployments horizontally by replicating application components such as application servers and data stores (Mather et al. 2009).
- *Pay-for-use* - Parties only pay for the services they consume. Parties do not need to buy any developing tools or full year license.
- *Transparency* - The PaaS provider manages upgrades, patches, and other maintenance as well as the infrastructure. The user does not need to worry about compatibility issues between the server configurations and the development software.
- *On-demand services* - PaaS providers offer software development tools that can be used by developers when needed.
- *Accessibility* - PaaS services are accessed through the Internet via web browsers from anywhere at any time.
- *Testability* - The variety of tools that we can use makes testing application programs in this environment more convenient.
- *Versatility* - PaaS offers different programming languages and databases. For instance, using Microsoft Azure, you can build applications using .Net, Java, PHP and others.
- *Simplification* - Developers do not need to buy or install any development tool, or to keep the servers updated. The applications are managed and maintained by the PaaS providers.

The Platform-as-a-Service pattern has the following liabilities:

- PaaS providers usually offer their own proprietary development software which makes hard to migrate an application from one vendor to another one. Also, APIs from different providers vary, which raises portability issues as well.
- The availability of the PaaS products depends mostly on the Internet. Thus, those services are available as long there are network connections.
- A PaaS provider can either own or sub-contract the underlying infrastructure to an IaaS provider. In either case, the security or availability of PaaS services may not be assured.

3.8 Related Patterns

Cloud infrastructure Pattern describes the infrastructure to allow sharing of distributed virtualized computational resources.

- Misuse Patterns in (Hashizume et al. 2012) describe possible attacks to cloud environments, which may affect the security of PaaS.
- Cloud Computing: Platform as a Service (PaaS) Pattern (Nexof) describes execution environments for PaaS applications.
- Party (Fowler 1997) indicates that users can be individuals or institutions.

4. CONCLUSION AND FUTURE WORK

We have presented two patterns for cloud service models. The Cloud Infrastructure pattern describes the infrastructure that allows the sharing of distributed virtualized resources such as server, storage, and network. The Platform-as-a-Service describes the middleware which provides virtual environments for developing, deploying, and delivering applications online.

We will develop the missing service model: Software-as-a-Service, as well as a reference architecture for cloud environments. These models are intended to study the security of cloud systems; we have developed some misuse patterns (Hashizume et al. 2012) in order to understand cloud threats from the point of view of the attacker. We intend to extend a forthcoming security pattern catalog (Fernandez), to include new patterns as defenses.

5. ACKNOWLEDGEMENTS

We thank our shepherd Lior Schachter for his insightful and careful comments that significantly improved this paper.

REFERENCES

Amazon Web Services LLC, —Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/

Amrheim, D. Forget Defining Cloud Computing. http://soa.sys-con.com/node/1018801

Baun, C., and Kunze, M. 2009. Building a private cloud with Eucalyptus, In *Proceedingd of 5th IEEE International Conference on E-Science Workshops*, 33-38

Camargo, R., Goldchleger A., Carneiro M., and Kon F. 2006. The Grid Architectural Pattern: Leveraging Distributed Processing Capabilities. In *Proceedings of the International Conference on Pattern Languages of Program Design 5*, 337-356

Centre for the Protection of National Infrastructure. 2010. Information Security Briefing 01/2010 Cloud Computing.

Dosani, M. 2010. On Cloud Nine Through Architecture*. Journal of Object Technology*

Eucalyptus Systems, Inc, http://www.eucalyptus.com/

Google, Inc, https://developers.google.com/appengine/

Fernandez, E. B., and Sorgente, T. 2005. A pattern language for secure operating system architectures. In *Proceedings of the 5th Latin American Conference on Pattern Languages of Programs*, Brazil, August 16-19

Fernandez E. B., Hashizume, K., Buckley, I, Larrondo-Petrie, M. M., and VanHilst, M. 2010. *Web services security: Standards and products*, Chapter 8 in *Web Services Security Development and Architecture: Theoretical and Practical Issues*. Gutierrez, C. A., Fernandez-Medina, E., and Piattini, M. (Eds.), IGI Global Group. 152-177

Fernandez, E.B. *Security patterns in practice: Building secure architectures using software patterns*. To appear in the Wiley Series on Software Design Patterns

Fowler, M. 1997. *Analysis patterns – Reusable object models*, Addison-Wesley

Hashizume, K. and Fernandez, E. B. 2009. A Pattern for WS-Security. *First IEEE Int. Workshop on Security Eng.Environments*, Dec. 17-19, Shanghai, China

Hashizume, K, Fernandez, E. B., and Yoshioka N. 2012. Three Misuse Patterns for Cloud Computing. *In Security Engineering for Cloud Computing: Approaches and Tool*, IGI Global

HP Cloud Service, http://hpcloud.com/

IBM, http://www.ibm.com/cloud-computing/us/en/

Karger, P., and Safford, D. 2008. I/O for Virtual Machine Monitors: Security and Performance Issues. *IEEE Security & Privacy*, Sep./Oct., 16-23

Lawton, G. 2008. Developing Software Online With Platform-as-a-Service Technology. *Computer*, vol.41, no.6, pp.13-15, IEEE, June.

Mather, T., Kumaraswamy, S., and Latif, S. 2009. Cloud Security and Privacy. O´Reilly Media, Inc.

Microsoft, Windows Azure, http://www.windowsazure.com/en-us/

Nexof. Cloud Computing: Platform as a Service (PaaS). http://www.nexof-ra.eu/?q=node/669

Nimbus. http://www.nimbusproject.org/about/

OpenNebula. Project Leads. http://opennebula.org/

Salesforce (1). http://www.salesforce.com/

Salesforce (2). Force.com: A comprehensive look at the World's Premier Cloud-Computing Platform. http://www.developerforce.com/media/Forcedotcom_Whitepaper/WP_Forcedotcom-InDepth_040709_WEB.pdf

Salesforce (3). An introduction to the Force.com IDE. http://wiki.developerforce.com/page/An_Introduction_to_Force_IDE

Salesforce (4). An Introduction to Environments. http://wiki.developerforce.com/page/An_Introduction_to_Environments

Salesforce (5). About the Force.com Developer Edition Environments. http://wiki.developerforce.com/page/Developer_Edition

Salesforce (6). Secure, private, and trustworthy: enterprise cloud computing with Force.com. http://www.salesforce.com/assets/pdf/misc/WP_Forcedotcom-Security.pdf

The National Institute of Standards and Technology (NIST). Cloud Computing Use Cases. http://www.nist.gov/itl/cloud/use-cases.cfm

Ubuntu. UEC Package Install Separatell. https://help.ubuntu.com/community/UEC/PackageInstallSeparate#Overview

Wang, L., Tao, J., and Kunze M. 2008. Scientific Cloud Computing: Early Definition and Experience. In *Proceedings of the 10th IEEE International on High Performance Computing and Communications*, 825-830