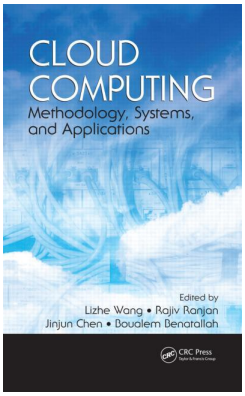
This article was downloaded by: *10.2.98.160*
On: 28 Oct 2020
Access details: *subscription number*
Publisher: *CRC Press*
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: 5 Howick Place, London SW1P 1WG, UK



Cloud Computing
Methodology, Systems, and Applications
Lizhe Wang, Rajiv Ranjan, Jinjun Chen, Boualem Benatallah

A Taxonomy of Interoperability for IaaS

Publication details
https://test.routledgehandbooks.com/doi/10.1201/b11149-5
Ralf Teckelmann, Anthony Sulistio, Christoph Reich
Published online on: 03 Oct 2011

How to cite :- Ralf Teckelmann, Anthony Sulistio, Christoph Reich. 03 Oct 2011, *A Taxonomy of Interoperability for IaaS from:* Cloud Computing, Methodology, Systems, and Applications CRC Press Accessed on: 28 Oct 2020
https://test.routledgehandbooks.com/doi/10.1201/b11149-5

PLEASE SCROLL DOWN FOR DOCUMENT

Full terms and conditions of use: https://test.routledgehandbooks.com/legal-notices/terms

This Document PDF may be used for research, teaching and private study purposes. Any substantial or systematic reproductions, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The publisher shall not be liable for an loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

3

A Taxonomy of Interoperability for IaaS

Ralf Teckelmann
Department of Computer Science
Hochschule Furtwangen University, Germany

Anthony Sulistio
Department of Applications, Models and Tools
High Performance Computing Center Stuttgart (HLRS), Germany

Christoph Reich
Department of Computer Science
Hochschule Furtwangen University, Germany

CONTENTS

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

The idea behind cloud computing is to deliver Infrastructure-, Platform- and Software-as-a-Service (IaaS, PaaS and SaaS) over the Internet on an easy pay-per-use business model. However, current offerings from cloud providers are based on proprietary technologies. As a consequence, consumers run into a risk of a vendor lock-in with little flexibility in moving their services to other providers. This can hinder the advancement of cloud computing to small- and medium-sized enterprises. In this chapter, we present our work in outlining the motivations and current trends of achieving interoperability, especially in the area of IaaS. More specifically, this work delivers a comprehensive taxonomy as a guideline for cloud providers to enable interoperability within their cloud infrastructures. Thus, this taxonomy discusses important topics of IaaS, such as access mechanism, virtual appliance, security, and service-level agreement.

3.1 Introduction

According to National Institute of Standards and Technology (NIST), there are five essential characteristics of cloud computing, i.e., *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity* and *measured Service* [509]. *On-demand self-service* means that customers are able to obtain computing capabilities without human interaction from a service provider. *Broad network access* defines the need for network-based access and standardized mechanisms in order to facilitate access through heterogeneous platforms.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

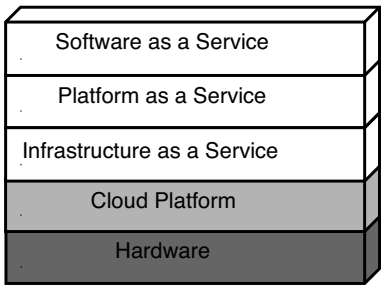


FIGURE 3.1
Layers of Cloud Computing.

Resource pooling describes providers' resources as pools of different physical and virtual resources dynamically assigned to consumers. The assignment and reassignment are based on a multi-tenant model in order to achieve high utilization. The possibility of rapid scaling up or down of provisioned capabilities is referred to as *rapid elasticity*. Moreover, the impression of infinite resource capabilities that are obtainable in a large quantity at any time is proposed. Finally, *measured service* means resources are automatically monitored, controlled, and if needed, optimized using metering mechanisms appropriate to the resource type. Furthermore, the according utilization is transparently traceable for customers and cloud providers.
The service models of cloud computing consist of *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)* and *Software as a Service (SaaS)* [509], as shown in Figure 3.1. They differ in the complexity of the provisioned capability, and how customers can access or use them. IaaS is marked through the provision of basic computing capabilities for customers. In PaaS, users have control only of their applications, and to a certain extent the configuration of hosting environments. In the SaaS model, customers just use a certain application running on a cloud infrastructure.

3.1.1 Motivation

Currently, offerings from first generation clouds, such as Amazon Web Services (AWS) [63], Rackspace [594], and Flexiant's FlexiScale [273], are based on proprietary middleware [588], thus, resulting in isolated environments. This isolation obstructs the further advancement of cloud computing. New models, e.g., federations, are not feasible without interoperability [611]. The same applies to the enforcement of Service-Level Agreements (SLA) in a hybrid cloud and the management of logically-grouped virtual machines (VMs).
The aforementioned issues lead to a series of disadvantages for customers, such as vendor lock-in and high migration costs. To address these, standardization efforts have to take place in order to support further developments in the

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

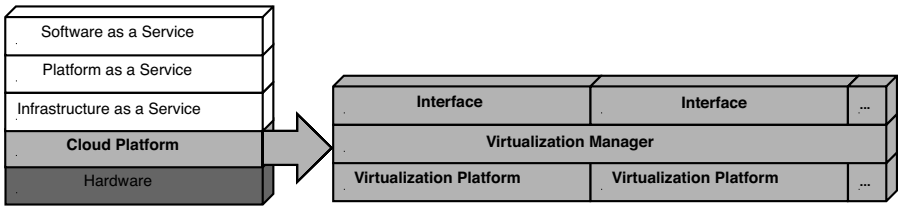


FIGURE 3.2
The Cloud Platform Layer.

second generation of clouds. Standardized exchange mechanisms and interfaces are crucial in order to facilitate interoperability [490]. It is also important that the standardization process should apply to the evolution of cloud.

As cloud computing becomes popular, myriads of offerings are announced. However, the first wave of these offerings are IaaS-based products like Amazon Elastic Compute Cloud (EC2) [63], Rackspace and FlexiScale. PaaS and SaaS solutions are later offered on top of IaaS. Figure 3.2 illustrates this relationship, which includes the *Cloud Platform* layer and its underlying hardware. Deriving from this stack, interoperability for IaaS has to be achieved first. In order to do so, standardization efforts have to focus on one layer below, i.e., the cloud platform layer, since it encapsulates all technologies used to provide virtual environment and provides functionalities to the upper layers. As shown in Figure 3.2, the cloud platform layer consists of several *Virtualization Platforms* that are responsible for the resource virtualization, a *Virtualization Manager* for managing VMs, and *Interfaces* that provide a controlled and uniform access to the underlying environment.

This chapter deals with technologies and mechanisms of the cloud platform layer specific to IaaS, in order to achieve interoperability between clouds. Only when the interfaces, documentation and standards are open and transparent, cloud services can be easily migrated to various platforms. To give an overview of related topics and important areas, this work presents a taxonomy. The taxonomy discusses all important issues to outline the needs and trends in current developments aiming for interoperability for IaaS. Important developments, such as the rise of Virtual Appliances (VAs) over VMs and the adoption of SLAs to clouds, are also considered in this taxonomy.

The rest of this chapter is organized as follows. In Section 3.2, the term *interoperability* is defined and discussed. Furthermore, the benefits of interoperability are pointed out. The core of this work is a taxonomy about IaaS interoperability and is presented in Section 3.3. Section 3.4 provides some related work. Finally, Section 3.5 concludes the chapter and gives future work.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

3.2 Interoperability of Cloud Platforms

According to Oxford Dictionary of Computing, interoperability refers to "The ability of systems to exchange and make use of information in a straightforward and useful way; this is enhanced by the use of standards in communication and data format" [215]. This definition applies well to cloud computing, since it addresses the current state of cloud solutions. There are multiple systems or clouds that are able to use the information that they have, but are not able to 1) exchange and share them, and 2) understand the information from others. Only the second point is somewhat related to interoperability.
There are two kinds of interoperability, i.e., *syntactic* and *semantic* [728]. Syntactic means that clouds are trying to communicate through standardized mechanisms, where interfaces, data formats and communication protocols are used to achieve interoperability. Semantic interoperability means that the information is not only exchanged but also interpreted in order to use it.

3.2.1 Benefits of Interoperable Clouds

The benefits of interoperability cover business/economy and technical issues. These issues are relevant to stakeholders, i.e., customers, developers, and cloud providers. From a business perspective, advantages to customers are: (i) no vendor lock-in, where a customer is no longer restricted to a single cloud provider; (ii) flexibility, where a customer is able to interact with others using various cloud offerings and distribute his/her applications across several providers; and (iii) cost reduction, where if a customer moves to a different provider, he/she can move major parts of the application without building it from scratch again. From a technical perspective, developers can take many advantages of using interoperable clouds:

- Monitoring of a distributed infrastructure scaled over several clouds can be achieved through a standardized communication mechanism.
- Consistent and uniform system management of multiple clouds. This addresses the current lack of having multiple information points, e.g., a local monitor for a private cloud, and Amazon CloudWatch [63] for virtual machines in the Amazon EC2.
- Secure verification and service discovery using standardized interfaces.
- SLA enforcement for the possibility of automatically reacting to a SLA violation.
- Fault tolerance by scaling over several clouds.
- Disaster recovery, where distributing information and components prevents complete data loss and reduces downtime.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

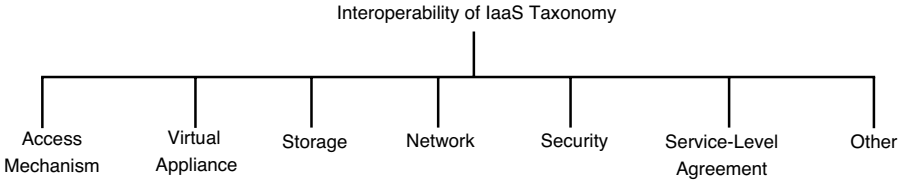


FIGURE 3.3
Interoperability of IaaS Taxonomy.

The above advantages provide cloud providers with a case for interoperability. One incentive for cloud providers is an increase in market share, as interoperability further drives wide-spread adoption by users and developers in using cloud computing. This translates to a growth in revenue. Another factor is to be able to provide users with a SLA guarantee, in case of hardware failure and/or security attacks. Using a federated cloud model as an example, users' VMs can be migrated to a different provider with minimal overhead, thus, reducing SLA violations.

3.3 Taxonomy of Interoperability for IaaS

Figure 3.3 gives an overview of the taxonomy categories, which can be seen as essential building blocks towards an interoperability for IaaS on the cloud platform level. The first category shown in Figure 3.3 is *Access Mechanism*, where several access types and important topics of service discovery are discussed. *Virtual Appliance* is the second category and deals with the interoperability of computational resources. Next is *Storage*, where it discusses storage organization and management as important areas for interoperability.
The fourth category shown in Figure 3.3 is *Network* that focuses on addressing issues regarding to high-level communications. Afterwards, *Security* is considered in this taxonomy. *Service Level Agreement (SLA)* is discussed next, especially in the area of architecture, template format, and monitoring of SLA objectives. Finally, the *Other* category is intended for topics like consensus and regulations, and audit standards that do not belong to other parts.

3.3.1 Access Mechanism

Access mechanisms have a major influence on interoperability since they define how a service can be accessed. Furthermore, they define how a service can be discovered. Two areas in access mechanisms are identified, i.e., *Types* and *Service Discovery*, as shown in Figure 3.4.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

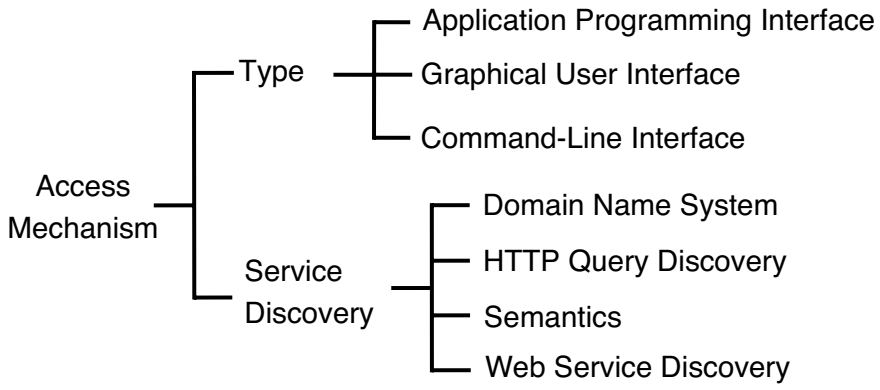


FIGURE 3.4
Access Mechanism Taxonomy.

3.3.1.1 Type

In an access mechanism, *Type* means different ways that a customer has access to services. In the first generation of clouds, common standard types are *Application Programming Interface (API)*, *Graphical User Interface (GUI)*, and *Command-Line Interface (CLI)*, as depicted in Figure 3.4.

Application Programming Interface

APIs offered by cloud providers can be divided into three architectural styles, i.e., *Web Service*, *HTTP Query*, and *Extensible Messaging and Presence Protocol (XMPP)*. Simple Object Access Protocol (SOAP) is a commonly-used specification for implementing web services, since it is a stateless message exchange protocol. The common underlying protocol is Hypertext Transfer Protocol (HTTP), but Remote Procedure Call (RPC) is also applicable. However, SOAP does not provide its own syntax within Web Services Description Language (WSDL), an XML-based language that provides a model for describing web services. Thus, it may lead to existing systems unable to communicate with each other due to using proprietary or implementation-specific solutions. HTTP Query provides an alternative mechanism to SOAP, where it uses either Remote Procedure Call (RPC) or Representational State Transfer (REST). Common RPC analogs are XML-RPC and JSON-RPC. XML-RPC is the precursor of SOAP and it uses XML to encode its calls, whereas JSON-RPC uses Java Simple Object Notation (JSON) as the data format. In contrast, REST is an architectural style. Although REST is exhaustively elaborate and explained in [271], no concrete definition is formalized. Thus, it leads to various interpretations on what RESTful means. This is reflected through APIs from

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

several cloud providers, such as Amazon and Rackspace that are promising RESTfulness, but unable to communicate with each other.
XMPP is an XML-based protocol. The design goals of XMPP are messaging, presence and request-response services in a near real-time manner. Like other XML-based communications over HTTP, XMPP can be adopted to a programmatic client-to-cloud communication. However, XMPP suffers from the low adoption by cloud applications.

Command Line Interface

Many cloud providers offer tools and scripts to support CLI, e.g., Amazon's EC2 API tools or other third-party tools for AWS. In most cases, these tools are just implementations on top of existing APIs like REST or XML-RPC. Secure Shell (SSH) also becomes a standard tool for accessing Unix/Linux-based computational resources.

Graphical User Interface

The main role of GUI is to provide users with an easy access to a cloud offering. In general, access using GUI can be differentiated into *Remote Desktop Protocol (RDP)* and *Web Portals*. RDP is a proprietary protocol from Microsoft, and it is used to connect VMs that use the Windows operating system (OS) remotely. In contrast, web portals are used independent of OS. Thus, many cloud providers like Amazon and Flexiant offer access to their IaaS offerings via web portals.

3.3.1.2 Service Discovery

Figure 3.4 also shows various forms of service discovery like *Domain Name System (DNS)*, *HTTP Query*, *Semantics* and *Web Service Discovery*. These forms are discussed next.

Domain Name System

DNS Service Discovery (DNS-SD) describes a mechanism to discover services of a certain domain [182]. It is a convention for naming of resource-record types, thus, allowing the discovery of service instances under a certain domain with basic DNS queries. DNS-SD provides a discovery mechanism, which can be integrated into existing DNS structures without major changes. Thus, DNS-SD is an access mechanism independent of a discovery mechanism.

HTTP Query

It refers to the *HTTP Query* described in Section 3.3.1.1. If a web service follows the REST constraint of statelessness [271], a discovery request on a resource should return all related information including the available operations. Moreover, each service is self-describing and discoverable through simple

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

HTTP requests. However, it only refers to the discovery of a service's capabilities, not the service itself.

Semantics

It aims to discover services by finding a meaning in certain contexts. Resource Description Framework (RDF) and Web Ontology Language (OWL) are commonly-used semantic models. RDF is a general-purpose language for representing information on the Web, and uses XML for the information presentation. In contrast, OWL enables the description of connections between information. Thus, resources can be described using RDF and their relations through OWL in a machine-readable way. As a result, resources can be discovered not only by keywords, but also by meaning. This makes service discovery very flexible, because applications are no longer bound to or developed against a static set of particular services.

Web Service Discovery

There are four aspects to a web service discovery, i.e., *Registry*, *Federated*, *Language*, and *Multicast*. Universal Description, Discovery & Integration (UDDI) is an XML-based registry for classifying and locating web service applications. It uses WSDL for service description and SOAP for communication.
A federated discovery is a hybrid version of distributed and centralized ones [660]. Several autonomous discovery entities are federated, and each of them provides its information within the federation.
Web Service Inspection Language (WSIL) is a language that defines formats and rules on how information about services are made available [113]. A WSIL document abstracts different descriptions' formats through a uniform XML-based format. It provides a mapping of different descriptions to the according service through pointers. Because the WSIL document only holds references and no concrete descriptions, it is extensible and easy to process.
Web Services Dynamic Discovery (WS-Discovery) is a multicast discovery protocol for locating web services in local area networks [556].

3.3.1.3 Discussion

CLI and GUI are important tools for giving users access to cloud offerings, but they lack interoperability. The only exception is standard tools for remote access to computational resources, which are widely-accepted like SSH or RDP. CLI and GUI are mainly implemented on top of existing APIs.
These APIs are the main access points. They use proprietary XML, JSON or plain HTTP as data formats resulting in a high diversity of implementation details. To achieve interoperability for IaaS, best practices or other restrictive rules are necessary, especially when using XML or JSON to describe data, due to high extensibility and adaptivity. Thus, a consensus about the meanings of syntactic constructs has to be achieved. Furthermore, the appropriateness

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

of existing approaches can be evaluated, and the inappropriate ones can be rejected in order to reduce the diversity.

In order to facilitate interoperability, APIs have to expose service discovery functionalities. A well-accepted approach is the combination of REST and HTTP Query service discovery of a certain domain. The advantages of this approach are ease of integration and simple discovery through well-known and well-defined technologies.

A federated service discovery mechanism can also help interoperability. A federation of mediators like WSIL offers the ability to encapsulate a broad band of services and offer request specific response. The approach of mediators also applies well with a semantic service discovery mechanism. An environment with services described in RDF is summarized using WSIL. Moreover, it can be discovered through the use of OWL, since it leverages the power of a semantic discovery, thus, leading to a more dynamic and flexible way. These technologies are also applicable in a RESTful architecture.

3.3.2 Virtual Appliance

The idea of *Virtual Appliance (VA)* is to deliver a service as a complete software stack installed on one or more VMs, as proposed by a Distributed Management Task Force (DMTF) [240]. A VA description is more than the XML-based summary of VM descriptions. Figure 3.5 shows the taxonomy of VA, where it consists of *Life Cycle*, *Virtualization Platform* and *Virtualization Manager*.

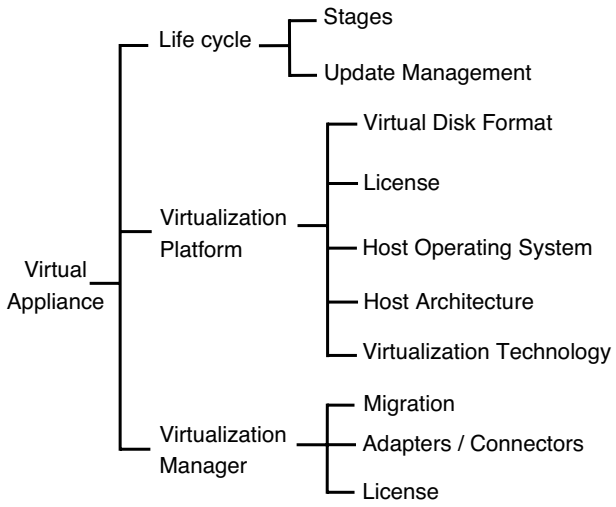


FIGURE 3.5
Virtual Appliance Taxonomy.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

FIGURE 3.6
The Life Cycle of a Virtual Appliance [240].

3.3.2.1 Life Cycle

As shown in Figure 3.5, there are two important topics of the VA life cycle, i.e., *Stages* and *Update Management*.

Stages

According to DMTF, the life cycle of a VA consists of five stages, as shown in Figure 3.6. In this figure, it starts with the *Development* stage, where images are prepared and meta-data are assembled. Next is the *Package and Distribute* stage. Because VAs consist of multiple entities, they have to be assembled. However, packaging means nothing more than a collection of all components. The *Distribute* part is the step of making the package available by uploading it to the cloud. Then, the *Deploy* stage launches the VA. The *Management* stage manages the VA, such as performing pause, resume, and stop operations. Finally, in the *Retirement* stage, the VA is decommissioned and the relevant resources are released.

Update Management

Currently, running VMs need to be updated individually by the users for software upgrades or security fixes. The update management can also pose a significant problem to cloud providers, as they need to update all pre-configured images. For PaaS and SaaS providers, an update management needs to be done carefully so as not to avoid SLA violations that disrupt VMs' uptime and network performance. Therefore, an update management can become a complex problem. Currently, one viable solution is to move from a conventional software patching to a redeployment [681]. With this solution, the update procedure is being shifted into the development phase of the VA life cycle. Thus, VA developers or maintainers can update their images and test them before they are being packaged and deployed. As a result, the newly-deployed VA takes over the older one after its retirement or slowly replaces it.

3.3.2.2 Virtualization Platform

Figure 3.5 also shows the core characteristics of a virtualization platform, i.e., *Virtual Disk Format*, *License*, *Host Operating System (OS)*, *Host Architecture*, and *Virtualization Technology*. Descriptions of each type are described next.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

Virtual Disk Format

The current landscape of virtual disk formats (VDFs) is heterogeneous, like VMware's Virtual Machine Disk (VMDK), Microsoft's Virtual Hard Disk (VHD), and QEMU's qcow. However, most of them are proprietary. To advocate interoperability, the support of various VDFs is recommended.

License

Software license is one of the most complicated topics because there are many free and proprietary licenses existing nowadays. Virtualization platforms with a free license are preferred in order to support interoperability. Moreover, they are less restrictive, hence, enabling broader adoption and agreement.

Host Operating System (OS)

In terms of interoperability, it is important that the host OS supports VMs with a variety of guest OSs, if a virtualization manager is not capable of managing different platforms. Moreover, the host OS should allow paravirtualization of a guest OS to enhance the performance, in comparison to using full virtualization.

Host Architecture

Several host architectures were developed within the evolution of computing like SPARC (Scalable Processor ARChitecture) and x86. The x86 architecture with 64-bit registers (x86-64) is compatible to 32-bit ones, but not vice versa. Since a VM's guest architecture depends on the host architecture and most x86 processors are used in new PCs and servers, x86-64 should be the standard host architecture to achieve interoperability.

Virtualization Technology

It is an essential part of virtualization, because it enables the running of multiple VMs inside one host or server. Currently, there are four technologies used in today's virtualization, i.e., full, para, OS-level, and hardware-assisted. Full virtualization allows a VM to be running without any OS modifications, for example, a VM with Microsoft Windows OS running inside a Linux-based host OS. In contrast, paravirtualization modifies a guest OS to communicate with a host system to improve performance and efficiency. Hence, VMs and the host need to be running on the same OS.
OS-level virtualization has the appearance of a stand-alone system from the view of running applications. However, the host OS is actually shared by many guest OSs. Hardware-assisted virtualization is a hardware technology that supports virtualization by enhancing the communication between a guest OS and the underlying hardware. For example, Intel's VT and AMD's AMD-V processors have built-in support to enable a hardware-assisted virtualization.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

To support interoperability, it is recommended that the cloud providers use a variety of virtualization technologies to address the different needs and requirements of their customers.

3.3.2.3 Virtualization Manager

A virtualization manager is responsible for managing VAs on physical hosts. For interoperability, an important task is to move the VAs to different hosts and/or cloud providers. Therefore, *Migration* is an essential task for the manager, as shown in Figure 3.5. Moreover, this figure shows other relevant interoperability issues like *Adaptors/Connectors* and *License*.

Migration

There are two kinds of migration, i.e., *cold* and *live*. For a cold migration, the VM is being shutdown and then moved to another host or server. This is a time- and resource-consuming process, where it has many disadvantages like increasing downtime and potentially violates SLAs. A second approach is a live migration that aims to avoid the aforementioned disadvantages. With this process, data need to be stored externally before the migration begins.
Some virtualization managers, such as VMware vSphere and Microsoft Hyper-V, are under closed source development and have restrictive licenses. Thus, they limit the ability to reproduce the same migration procedures and perform uniform benchmark tests [191]. Moreover, with various vendors using different live migration strategies, it would be a challenging task to achieve interoperability in this issue.

Adaptors/Connectors

The terms adaptor, connector, broker or driver are refer to the same thing, i.e., the integration of an API in a modular way that allows the translation of operations and commands from one API to the function set of another. This modular approach is adopted by multiple virtualization managers of second generation clouds like OpenNebula [667] and Eucalyptus [544].
The modular approach results from the need to achieve a certain level of interoperability through the integration of a variety of interfaces. Currently, only the proprietary Amazon EC2 API is gaining wide acceptance since interfaces from Open Grid Forum (OGF) [551] and DMTF are still in an early adoption state. However, in the future, the adoption of several APIs through adaptors should be a de facto requirement for a virtualization manager.

License

The same issues with a license for a virtualization platform, as discussed in Section 3.3.2.2, can be applied to a virtualization manager.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

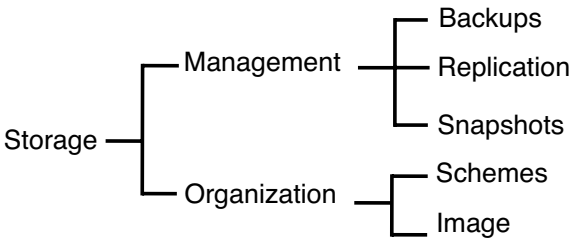


FIGURE 3.7
Storage Taxonomy.

3.3.2.4 Discussion

In this chapter, virtual appliances are introduced and presented as a new computational resource for cloud computing, instead of virtual machines. As mentioned earlier, the VA approach provides new possibilities. Thus, its adoption and further enhancement is recommended.
Due to heterogeneity in virtual disk formats, licenses, host and guest OSs, and virtualization technologies, only host architecture and virtualization manager can be used to achieve interoperability. The virtualization manager is capable of abstracting various virtualization platforms through an intelligent controller, which orchestrates the platform utilization and allows a heterogeneous environment. However, interoperability among virtualization managers is obscured by closed or proprietary sources, and license restrictions. Interoperability can only be achieved through the adoption of virtualization platforms and virtualization managers without such limitations. The adaptors/connectors concept can also help interoperability though a modular integration of various APIs.

3.3.3 Storage

Figure 3.7 shows two important topics in terms of storage interoperability, i.e., *Management* and *Organization*. The former addresses storage functionalities and discusses the need of their availability. The latter points out several kinds of storage organization, which are important to prevent incompatibility.

3.3.3.1 Management

Figure 3.7 also shows the three topics of a storage management, i.e., *Backup*, *Replication*, and *Snapshots*. Because these topics are sometimes mixed up or treated as alternatives, their differences are highlighted to avoid this confusion. Moreover, they have to be present in order to achieve interoperability of IaaS. The backup functionality is important to guarantee long-term preservation on a non-volatile storage media, either for data to be backed up or VM images

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

to be protected against data loss. On the other hand, replication means that data are not only being copied to one location, but they are distributed to several sites. However, replicated data are not moved into a non-volatile storage media. Instead, they are copied to volatile storage for availability and faster access time. Finally, snapshots are full copies of data objects. Compared to backups, snapshots are not moved to non-volatile storage. Snapshots are also different from replications, because in the process only one copy is made to a single location and not to multiple ones.

3.3.3.2 Organization

Storage organization depends on the kind of data to be stored. Figure 3.7 also shows the areas of a storage organization, i.e., *Schemes*, and *Image*.

Schemes

There are three major schemes of storage organization: *Block Storage*, *File System*, and *Object Storage*. Block storage is a kind of storage where data are saved in blocks. A block has a certain size and consists of a structured sequence of bytes. Thus, block storage can be seen as a logical array of unrelated blocks, which are addressed by an index in the array called Logical Block Address (LBA) [260]. On the other hand, a file system is responsible for imposing a structure on the address space of disks, such that applications can refer to files' abstract names of data objects. Finally, object storage is a kind of abstract storage consisting of one or more object stores or devices [260]. An object store is a collection of objects, which consists of data objects and their metadata. With this approach, objects can be stored or modified via methods exposed through a standardized file-like interface. Everything below the method calls is encapsulated.

Image

Image formats have already been discussed in Section 3.3.2.2. Hence, this section focuses on their relationship to the underlying organization scheme. Hard disk images are one example already mentioned above. A hard disk image is a by-sector copy of a hard disk. The important point is that sectors are smaller than blocks, different from files, which are mostly larger. Therefore, the image could be placed on the block storage more efficiently. Furthermore, all information is within the image and no metadata have to be considered. In case of a VA image containing files [241], the usage of block storage is not efficient due to additional file sources.
Decoupling pairs of images and related data to the most appropriate scheme can be a solution, but only possible if the schemes are hidden behind intelligent object storage. In this scenario, block and file system storages are chosen based on the type of data. Therefore, interoperability can be achieved through the intelligent placement of data beneath an object storage layer.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

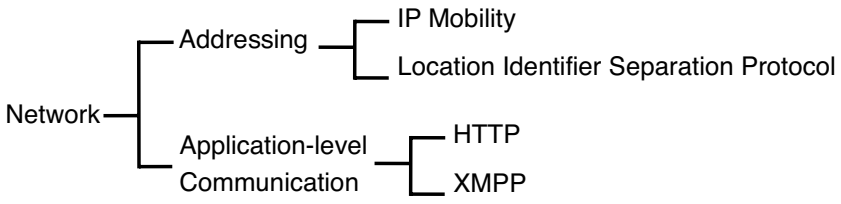


FIGURE 3.8
Network Taxonomy.

3.3.3.3 Discussion

Compatibility of underlying storage resources is required to allow the shift of data from one location to another while still retaining the possibility of an efficient usage. This issue is highlighted through the introduction of three storage organization schemes, and the statement about the relationship between various images and the underlying storage infrastructure. To avoid compatibility problems, the introduction of an intelligent object storage layer can be a solution. VM images and related data are decoupled and stored on most appropriate storage. Logical connections can be kept using metadata information. The mentioned functionalities can be supported from such an environment and fulfill interoperability requirements.

3.3.4 Network

Figure 3.8 shows two important topics in terms of network interoperability, i.e., *Addressing* and *Application-level communication*.

3.3.4.1 Addressing

A major problem for cloud computing is the need to have a reliable remote access to applications and their underlying VMs, even when they are being moved between subnets within a cloud or to others. Therefore, network addressing plays an important role for interoperability. As shown in Figure 3.8, *IP Mobility* and *Locator Identifier Separation Protocol (LISP)* aim to address the aforementioned problem.

IP Mobility

The possibility to keep its IP address during live migration is essential for a VM, otherwise it would directly lead to service downtime and SLA violations. Therefore, extended mechanisms need to be considered to allow seamless migration and facilitate interoperability. The mechanisms of an IP mobility are available for both IPv4 and IPv6.
IPv4 mobility is an extension of the IPv4 protocol [247]. The first component

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

is the so called a *home agent*. This can be a network router that a VM is working on. When the VM moves, a so-called *care-of address* of the machine is deposited to the home agent. This is the address through which the VM is accessible in the new network. If the address is provided by the router, then this router is called *foreign agent* and has the function to forward data to the VM. When datagrams are sent to the migrated VM, the home agent tunnels them to the foreign agent. In the other direction, the foreign agent serves as a standard gateway.
IPv6 mobility implements a mechanism similar to IPv4. Care-of addresses and home agents are also used [398], but there are many differences resulting in incompatibility and the inability to inter-operate. The most important difference is that, because of mechanisms like proxy neighbor discovery and stateless or stateful auto-configuration, IPv6 mobility does not need foreign agents [398]. Furthermore, the communication routes differ, because IPv6 mobility allows bidirectional tunneling over the home agent, as well as direct communication to the care-of address for the other communicating entity.

Locator Identifier Separation Protocol

LISP aims to have a single IP address to be used for multiple purposes [126]. An IP address locates and identifies a host. The idea is to separate these functionalities into *Endpoint Identifiers (EIDs)* and *Routing Locators (RLOCs)* [263], where EIDs are hosts and RLOCs are router addresses. End-to-end communication is realized on EID-to-EID communication. EIDs are mapped to their according RLOCs through conversion algorithms resulting in a RLOC-based communication on the way between start and destination network. Therefore, LISP is placed on edge routers. This model works with both IPv4 and IPv6, and allows mobility over network barriers through the decoupling of ID and location. However, because of the placement of LISP on edge routers, the implementation and adoption is extremely difficult even if the model seems to be powerful. Deriving from this, further research is needed.

3.3.4.2 Application-Level Communication

Figure 3.8 also shows two communication protocols important in terms of interoperability, i.e., *HTTP* and *XMPP*. HTTP was already mentioned in the context of RESTful APIs in Section 3.3.1.1. However, in cloud computing, HTTP is the common application-level transport protocol, since HTTP mechanisms apply well to REST and SOAP.
XMPP has also been discussed previously, where it provides XML over TCP. Unfortunately, due to its low adoption in clouds and the deficiencies already pointed out in Section 3.3.1.1, XMPP is not an alternative to HTTP in terms of interoperability.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

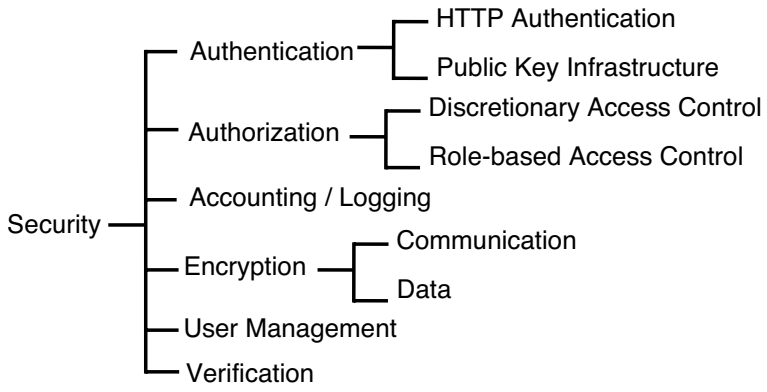


FIGURE 3.9
Security Taxonomy.

3.3.4.3 Discussion

Interoperability in networking is achieved through the dominant position of the IP protocol family. It is achievable even if IPv4 and IPv6 are incompatible, because there is cooperation among translation mechanisms. The ability of a continuous low-level communication is essential, but it is currently not possible among heterogeneous environments. A comprehensive solution through LISP does not seem feasible at the moment. Therefore, a homogenous addressing founding on IPv6 has to become standard in order to achieve an interoperability of cloud platforms. This is due to the exhaustion of the IPv4 address space, and the IPv6 mobility is a core component of the initial protocol design and has no later addition.
XMPP is currently not suitable for a high-level intercloud communication, since it is not widely-used by cloud applications. Therefore, HTTP can be seen as the only way to achieve interoperability in application-level communication. Nevertheless, the XMPP approach should be traced and reviewed against the applicability of REST to it.

3.3.5 Security

Figure 3.9 shows important security topics for a cloud interoperability. *Authentication*, *Authorization* and *Accounting/Logging* introduce the AAA model into this taxonomy. Next, *Encryption* mechanisms for communication and data are presented. Afterwards, important considerations in the area of *User Management* are pointed out. Finally, a description about a service discovery specific topic *Verification* is discussed.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

3.3.5.1 Authentication

Authentication means a confirmation of a stated identity, and it is an essential security mechanism in clouds and other IT areas. In the following, two kinds of mechanism are presented, as shown in Figure 3.9.

HTTP Authentication

HTTP offers two authentication mechanisms, i.e., basic and digest access [285]. The basic authentication is founded on a Base64 encoding of user ID and password within the authorization header field. It is not secured since Base64 is an encoding not an encryption mechanism. Digest access authentication addresses this problem through MD5-based encryption for the password and several other values, but not the username. Unfortunately, due to the well-known weakness of MD5, this mechanism can not be treated as secure. Moreover, these mechanisms are not interoperable with each other.

Public Key Infrastructure

Public Key Infrastructure (PKI) is a foundation for security mechanisms that allows the use of public keys and certificates, instead of logins and passwords [755]. Therefore, PKI provides an opportunity to establish a trustable relationship between an individual or an organization to its credentials (key or certificate), which can be used by other mechanisms to verify an identity and authenticate it. In terms of interoperability, four components of PKI are important, i.e., *Trust Model*, *Cross Certification*, *Algorithms*, and *X.509*.
There are several trust models available for PKIs. One model is the hierarchical model. There, one or more entities, called certification authorities (CAs), build a hierarchy above the credentials of an end-user. In this model, a CA guarantees the confidentiality of the CAs or end-user credentials below and can be verified against a next higher CA if necessary. Another model is called *Web of Trust*. Because no higher authorities are present, confidentiality has to be verified on each end-user characteristic. In case of Pretty Good Privacy (PGP), other end-users acknowledge an identity and the consumer decides if he/she trusts these acknowledgements. A third model is the *Peer-to-Peer* model were each entity establishes trust with each other entity through individual trust negotiation.
Cross certification is an extension of the hierarchical model. It gives entities of one PKI the chance to verify the confidentiality of entities belonging to other PKIs. The idea is quite simple; if one CA trusts another, an entity trusting one of them can also trust the other one.
In cryptography, the requirement on replacing algorithms is recommended, because of the need to be able to substitute an algorithm with a stronger one if it got broken. Thus, it is necessary that communicating systems can negotiate an algorithm they are capable of. On the other hand, X.509 is an International

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

Telecommunication Union (ITU) standard for important PKI components like public key certificates, certificate revocation lists, and attribute certificates.

3.3.5.2 Authorization

Authorization means an allocation of resources or rights according to the user's credentials after the user has proven to be the one he/she stated. Mainly two models, *discretionary* and *role-based access control* are discussed, as shown in Figure 3.9.

Authorization in clouds is achieved through Discretionary Access Control (DAC). Amazon and Rackspace are examples of using DAC in their public clouds. The concept of separation of users and objects hides behind this term [232]. The access in this case is controlled through lists, named Access Control Lists (ACLs). These lists consist of mappings of usernames and functions that they are able to execute on the according object. The mappings are determined by the owner of the object. This model has a great potential for interoperability, because only the ACL format has to be uniform to achieve interoperability. The function terms do not have to be unified, since they are only applied to a specific environment.

Role-based Access Control (RBAC) is based on the assumption that access control decisions are determined by the role of a user [266]. This can be by his/her duties, responsibilities or qualification. In RBAC, functions are dedicated to users according to their roles, resulting in a concept of different roles with different function sets. Unprivileged users, privileged users, database administrators or network administrators are examples of roles. Another difference from DAC is that these roles are not determined by users or owners, instead they are specified by the system. Currently, only one cloud API, VMware's vCloud with vExpress, implements the RBAC approach. A multi-tiered RBAC separates user and administrative functions. However, RBAC is not compatible with DAC, since the roles of RBAC systems can differ.

3.3.5.3 Accounting/Logging

In conjunction to security, accounting/logging means the record of the amounts of events and operations, and the saving of information about them. In terms of security, the importance of accounting data lies in their availability sustaining trust and compliance. Logs are central information sources for common system and fault analysis. Interoperability between clouds can be seriously affected if no such information is available and there are no present mechanisms to access them.

The concrete information could be made available using Information Technology Infrastructure Library (ITIL)'s *Incident Logging* [379]. This ITIL process defines several parameters which have to be present in every event record. Furthermore, it proposes multi-level categorization for the identification of event importance. Event-based logging would also apply well to a monitoring solution to be discussed later in Section 3.3.6.3. The influence of ITIL and its

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

well-defined best practice guidelines could lead to acceptance and adoption. As a result, trust and interoperability can be achieved.

3.3.5.4 Encryption

Figure 3.9 also shows two important topics of encryption, i.e., *Communication* and *Data*. Security between endpoints through communication encryption is as important as endpoint security. Especially through the close relationship of REST and SOAP to HTTP, Secure Socket Layer (SSL) or its new version Transport Layer Security (TLS), become the standard encryption protocol for communication over unsecure networks. Virtual Private Networks (VPNs) and SSH are other common mechanisms providing secure communication per default. For Windows-based systems, RDP also provides a secure mechanism. Data encryption is a topic mostly all cloud providers distance themselves from. Currently, the only agreement made is that existing encrypted data can be stored [350]. The problem of encryption methods is the lack of knowledge of their practicality in cloud computing, due to its scale. Mechanisms like `encfs`, Linux Unified Key Setup (LUKS) or `dm-crypt` are feasible for local systems with lower scale data size (in terms of Gigabyte), and are subjects of open licenses. However, their feasibility on cloud scenarios is not yet proven.

3.3.5.5 User Management

In the first generation of clouds, data regarding a user profile are mostly limited to login name, password, e-mail address and credit card number. The implementation of a secure authentication environment like a PKI raises the data amount as a direct consequence. This potentially enlarges the profile through certificates, stored public keys and signatures, and more personal or organizational data necessary for such security mechanisms.
A lot of operations in the cloud are based on user interactions that require credentials. Thus, how are profiles treated in case of federation? One solution is to leverage a Single Sign-On (SSO) technology with Shibboleth, where it allows a single user credential to access and use various services across multiple layers [657]. In Shibboleth, special components like *Identity Providers (IdPs)* and *Service Providers (SPs)* are introduced to build an environment which is also able to federate with other SSO systems. A user or system account has to authenticate if he/she wants to use a service, and has to communicate with the SP. Then, the SP verifies the provided credential against information it gets from the IdP. If the user is authenticated, credentials are placed on his/her side; then the SP can be used to authenticate user interactions without disturbing the user. The process can also be realized between SPs and IdPs of federated partners. Moreover, the amount of data retrieved between IdP, SP and the user can be configured, thus improving the security of the user data.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

3.3.5.6 Verification

Verification is a security requirement for the identification of services in conjunction with the service discovery [126]. Verification means the ability to decide if a service or application can trust others [350]. If it can not, then interoperability will be hindered, because services are not able to prove their intentions as stated. Thus, the communication will be rejected due to security issues. In order to avoid this, verification through certificates or signatures is needed. Every service of a domain owning a certificate or being signed with a verifiable signature can be assumed as trustable. For example, RDF or WSIL documents could be signed easily. Overall, verification introduces security into the service discovery process and into the interaction of communicating services. This strengthens interoperability as the requester can assure itself about the authenticity of its partner.

3.3.5.7 Discussion

HTTP authentication does not provide a secure mechanism, so it is not applicable commercially. PKIs, especially hierarchical ones that use well-known and secure mechanisms are recommended to achieve interoperability in the short run, and with the support of cross certification or SSO in the long run. X.509, as a standard certification format used in mostly all PKIs, also provides interoperability. Moreover, certificates can be used to verify more than just users. Services and service entry points can be provided with certificates as well.

Only DAC becomes widely accepted since RBAC plays a smaller role. Interoperability between DAC and RBAC is not achievable due to different underlying concepts. Even if RBAC provides several advantages compared to DAC, its adoption is not as common as DAC. This leads to interoperability through the broad adoption of ACL-based DAC for authorization.

Accounting/logging enables trust and compliance through the availability of event and operation traceability. Interoperability between logging facilities can be achieved through APIs providing information and hiding the concrete infrastructure. However, their presence is important to achieve the willingness to inter-operate. The functionalities exposed through APIs should follow ITIL guidelines, since these best practices are widely accepted.

The two topics, communication and data encryption, are also pointed out in this section. However, data encryption lacks adoption within cloud computing. User management in federated scenarios is also an important topic, because personal data have to be stored, but not transmitted. If an operation needs to be billed, information will be needed by the partners. This process can be arranged through the utilization of federated SSO infrastructures. The operating principle of Shibboleth and the resulting possibilities were highlighted previously. The proposed approach also applies well to the authentication and authorization mechanisms, because of the usage of common strong security and web standards. A critical point is the difference from the underlying

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

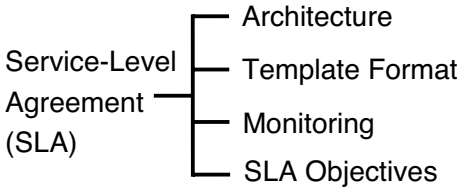


FIGURE 3.10
Service-Level Agreement Taxonomy.

attribute-based authorization. Further research efforts are necessary at this point.
Programmatic service discovery is another important feature for automated service delivery and consumption. Verification secures the communication between services through the introduction of verifiability of the confidentiality of their opponents. This strengthens the idea behind interoperable systems. Therefore, verification mechanisms have to be adopted in form of signing the service discovery endpoints with verifiable certificates.

3.3.6 Service Level Agreement

Service Level Agreements (SLAs) are ubiquitous in modern IT systems, and thus, they have to be discussed in terms of interoperability [126] [350] [606] [567]. Figure 3.10 shows four important topics, i.e., *Architecture*, *Template Format*, *Monitoring* and *SLA Objectives*.

3.3.6.1 Architecture

Web Service Agreement Specification (WS-A) is the standard for SLA management in web service environments, and its web service-based interface called *agreement layer* [87]. While WS-A is XML-based and exposes a rich function set, RESTful is preferred since it is minimal, with extensible function sets of Virtual Execution Environment Management Interface (VMI).

3.3.6.2 Template Format

Several template formats for SLAs were developed in the past years, for example IBM's Web Service Level Agreement (WSLA) or SLA Definition Language (SLAng). One important property is that these template formats are machine-readable documents, where they are the electronic representations for the purpose of automated management. In the first generation cloud systems, SLAs have predefined metrics and values for SLA Objectives (SLOs), and they are mostly not machine readable [513]. This results in one-round agreement, where a customer can accept or reject an agreement. The possibil-

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

ity to negotiate the metrics and values of service level objectives on runtime allows multiple round agreements and is more dynamic.
One template format developed in an extensible way with such scenarios in mind is WS-A, which is an OGF standard. Similar to WSLA and SLAng, WS-A offers a language to describe a SLA template and all of its components, and a basic XML-based schema for automated management. Moreover, WS-A specifies a protocol advertising the capabilities of service providers and creating agreements based on creational offers, and for monitoring agreement compliance at runtime [87].
In order to achieve interoperability, it is necessary to agree upon a single template format. The possibility to process documents electronically is crucial for automated management and the extensibility to dynamic negotiation is required also. WS-A seems to be a good choice; however, cloud providers have not yet started to offer machine-readable SLAs. Deriving from the choice of XML as the description language for WSLA, SLAng and WS-A, an XML-based template format seems to be in favor.

3.3.6.3 Monitoring

Monitoring of SLAs is important for both cloud providers and customers. A provider wants to ensure that the provision of resources is according to the SLA and no liability arises. On the other hand, a customer wants to ensure the adherence of cloud providers as mentioned in the contract. A way to monitor SLA is to use event-based dynamic monitoring infrastructures capable of associating low-level metrics with Key Performance Indicators (KPIs) of SLOs. Thresholds can be determined dynamically and re-negotiated between parties having already accepted them in case of events forcing a change [199].
In terms of interoperability, a dangerous scenario arises when high static thresholds of SLOs can not be satisfied by any cloud providers. The consequence is permanent SLA violations and the inability to move to other clouds, because the SLA requirements could not be met. Moreover, an insufficient event monitoring of static threshold monitoring systems, which are not capable of re-negotiation, can stress interoperability. The result are possibly more critical SLA violations from shift delays and unavailability. Overall, an event-based dynamic monitoring system fits better to the dynamic nature of clouds, dynamic SLOs, and machine-readable and negotiable SLAs.

3.3.6.4 SLA Objectives

SLA Objectives (SLOs) are the core component of a SLA, because they directly influence the interoperability of cloud systems. In a scenario where a customer wants to change his/her cloud provider, the absence of SLOs can make it difficult to compare the old SLA with a new one, for example. In order to achieve interoperability, dynamic negotiation and re-negotiation are in favor [199] [513] [87]. This approach applies well to a modular and independently-extensible SLA management entity and a dynamic extensible

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

machine-readable SLA template format previously discussed. For SLOs, dynamic negotiation means the participating parties communicate and negotiate SLOs, implied values, and related information, to which only relevant SLOs are of concern.

3.3.6.5 Discussion

APIs are crucial to interoperability for exposing SLA management functionalities. Furthermore, the separation of a SLA managing entity strengthens its interoperability through exchangeability and extensibility. In addition, a modular architecture that supports dynamic SLA negotiations avoids interoperability issues, since SLA terms can be negotiated during runtime. Assuming that machine-readable documents become common, a sole-accepted SLA template format is essential to achieve interoperable systems, since current existing formats are not compatible. WS-A seems to be a good choice, due to its extensibility through XML and orientation for more complex negotiation processes, and an OGF standard.
The danger in SLA monitoring for interoperability arises from characteristics of monitoring systems using a static low-level threshold to realize high-level SLOs. Not only permanent or more critical SLA violations can occur, depending on the monitoring systems behavior, the cumbering of interoperability through unsatisfied SLOs can be a consequence as well. Furthermore, dynamic monitoring systems apply better to the dynamic nature of clouds and other SLA-related mechanisms like dynamic SLO negotiation.
KPIs are well-known and should be present in cloud computing. However, mechanisms for dynamic negotiation are still required. Interoperability of SLOs can be achieved through the negotiation of KPI and all other relevant information and their aggregation to SLOs. Measures can be controlled by using a dynamic threshold-based monitoring system, which is encapsulated as a SLA service. This service again provides the necessary functionality in form of an API to the cloud provider and customers. Such a system would address all issues and achieve interoperability.

3.3.7 Other

Besides the mostly functional topics discussed for this taxonomy, other topics can have major input on interoperability as well. The topic on *Consensus* highlights the importance for conformity of initiatives, companies and communities. Some compliance issues with potential cumbering influence on interoperability are discussed in *Regulation and Auditing Standards*.

3.3.7.1 Consensus

As already pointed out and shown in the previous sections, standard compliant implementations and de-facto standard tools play a decisive role in cloud computing. Interoperability can be achieved if standards get adopted and certain

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

tools accepted. Standard Development Organizations (SDOs) and companies try to achieve this by founding on alliances and cooperation with other initiatives or companies. Open and academic communities are also part of these alliances. However, a broad consensus about standards and best practices is necessary between company- and community-driven projects to avoid the old battle of open source communities against closed source companies. The first generation of clouds lacked of interoperability, due to the absence of standards and best practices. The second generation should not fail, since differentiating business interests are resulting in the creation of a smaller number of still not interoperable cloud specifications for protocols and APIs.

3.3.7.2 Regulation and Auditing Standards

Regulations can also have major influence on interoperability, because governmental laws or regulations could restrict or prohibit interoperability. For example, storing data in the cloud may elicit various federal and state privacy and data security law requirements, such as the US Health Insurance Portability and Accountability Act, and EU Data Protection Directive [669]. Thus, privacy and data security laws present a significant challenge for cloud providers to comply with.

Auditing standards can also play important roles for interoperability [350]. For example, ITIL provides an auditable best practices catalog for IT Service Management (ITSM) [379]. However, in cloud computing, IT resources are no longer solely in users' own data center. Therefore, it is at the discretion of a cloud provider to follow standards.

3.4 Related Work

Other publications focus on specific problems, such as requirements [567], security [469], semantic-based [195], or mainly express some considerations [490] and use cases [239]. However, several publications analyze the cloud computing paradigm through the establishment of taxonomies [606] [588] [126]. This has the advantage to be able to structure the components and define their borders, comprehensiveness and influences on other topics.

Rimal et al. [606] focus on a high-level architectural observation of cloud functionalities and capabilities. They use these criteria to compare a few cloud providers and their offerings. However, they do not focus on IaaS and interoperability. This makes their taxonomy more general. Moreover, they focus only on existing cloud offerings, but not on current developments and standards.

Prodan and Osterman [588] also utilize a taxonomy to summarize the elements in a cloud environment, but from a general perspective. This is reflected in the analysis of several cloud providers and web hosting providers who are

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5

against their proposed taxonomy. Similarly, the difference to our work lies in the specialization on IaaS, interoperability and the analysis of different technologies.
On the other hand, Bernstein et al. [126] focus on an intercloud communication by discussing multiple problems and providing solutions to them. The result of their work is a set of protocols, which can be utilized to achieve an intercloud communication. In their work, the use of open standards is essential in order to achieve interoperability. However, this approach focuses only on the communication part, but does not consider current trends or other important topics, such as virtual appliance, security, storage and SLA.

3.5 Conclusion and Future Work

This work presented a taxonomy of important categories and topics in the area of interoperability for Infrastructure as a Service (IaaS). Firstly, a detailed explanation of cloud computing is presented. Then, the term interoperability is defined and elaborated, and its benefits to stakeholders, i.e., customers, developers and cloud providers, are explained.
The taxonomy itself spans many essential cloud computing topics, such as access mechanism, virtual appliances (VAs), security, Service-Level Agreements (SLAs), and general recommendations. Moreover, a comprehensive scope of topics is considered and hot topics are mentioned. The depth and width of the categories are varied, but this taxonomy outlines a detailed picture of the significant characteristics. However, there are several times mentioned in this chapter, where lack of current solutions are identified. Thus, further research is warranted.
As for future work, VA life cycle management, network addressing issues and SLAs need to be studied in more detail. Moreover, the use of the Information Technology Infrastructure Library (ITIL) to provide interoperability is an interesting area to explore. Finally, further work is needed to compare existing cloud standards and offerings with the topics and criteria presented in this taxonomy.

© 2012 by Taylor & Francis Group, LLC

Downloaded By: 10.2.98.160 At: 14:18 28 Oct 2020; For: 9781439856420, chapter3, 10.1201/b11149-5